4050 midterm project

```
library(tidyverse)
```

```
── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
✔ ggplot2 3.3.6      ✔ purrr   0.3.4
✔ tibble  3.1.8      ✔ dplyr   1.0.10
✔ tidyr   1.2.1      ✔ stringr 1.4.1
✔ readr   2.1.2      ✔ forcats 0.5.2
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
```

```
library(readxl)
library(ggplot2)
library(ggpubr)
library(GGally)
```

```
Registered S3 method overwritten by 'GGally':
  method from
  +.gg   ggplot2
```

```
library(caret)
```

```
Loading required package: lattice

Attaching package: 'caret'

The following object is masked from 'package:purrr':

    lift
```

```
library(ISLR2)
library(tidyverse)
library(dlookr)
```

```

Attaching package: 'dlookr'

The following object is masked from 'package:tidyr':

    extract

The following object is masked from 'package:base':
```

```
    transform
```

```
library(dplyr)
library(ggpubr)
```

1. DATASET PREPARATION

1.1 read files and merge:

read dropout, test, financial, static, progress and test files

```
df_label<-read.csv("Student Retention Challenge Data/DropoutTrainLabels.csv")
df_test<-read.csv("Student Retention Challenge Data/Test Data/TestIDs.csv")

## read financial files
df_financial <- read_xlsx("Student Retention Challenge Data/Student Financial Aid Data/20
## change colname and prepare for joining
names(df_financial)[1]="StudentID"
df_financial$StudentID <- as.numeric(df_financial$StudentID)

## read static file
static_data <- lapply(Sys.glob("Student Retention Challenge Data/Student Static Data/*.cs
## merge static data
df_static<-static_data%>%bind_rows
##  remove repeated column
df_static <- df_static %>% select(-c(Cohort, CohortTerm))

## read progress file
## Delete static variables & Mark semester and year for variables.
read_sp_file<-function(year,term,filename){
  address<-paste("Student Retention Challenge Data/Student Progress Data/",filename,sep="
  term_year_SP<-read.csv(address)
  term_year_SP<-select(term_year_SP,-2:-5)
  for (i in 2:13){
    names(term_year_SP)[i]=paste(colnames(term_year_SP)[i],term,year,sep = "_")
  }
  return(term_year_SP)
}

###  read fall
for (i in 2011:2016){
  assign(paste("Fall_",i,"_SP",sep=""),read_sp_file(year=i,term="Fall",filename=paste0("F

}
### read spring & summer
for (i in 2012:2017){
```

```
for (i in 2012:2017){
  assign(paste("Spring_",i,"_SP",sep=""),read_sp_file(year=i,term="Spring",filename=paste
  assign(paste("Sum_",i,"_SP",sep=""),read_sp_file(year=i,term="sum",filename=paste0("Sum
}
```

merge training set

```
df_train<-merge(x=df_label,y=df_financial,by="StudentID",all.x=TRUE)
df_train<-merge(x=df_train,y=df_static,by="StudentID",all.x=TRUE)
## merge based on Fall, Spring, Summer
## first merge Fall 2011
df_train <- merge(x = df_train, y = Fall_2011_SP, by = "StudentID", all.x = TRUE)
## next merge 2012:2016 Spring, Summer, Fall
for(i in 2012:2016) {
  df_train <- merge(x = df_train, y = get(paste0("Spring_", i, "_SP")), by = "StudentID",
  df_train <- merge(x = df_train, y = get(paste0("Sum_", i, "_SP")), by = "StudentID", al
  df_train <- merge(x = df_train, y = get(paste0("Fall_", i, "_SP")), by = "StudentID", a
}
## last merge Spring and Summer of 2017
df_train <- merge(x = df_train, y = Spring_2017_SP, by = "StudentID", all.x = TRUE)
df_train <- merge(x = df_train, y = Sum_2017_SP, by = "StudentID", all.x = TRUE)
```

merge test set

```
df_test<-merge(x=df_test,y=df_financial,by="StudentID",all.x=TRUE)
df_test<-merge(x=df_test,y=df_static,by="StudentID",all.x=TRUE)
## merge based on Fall, Spring, Summer
## first merge Fall 2011
df_test <- merge(x = df_test, y = Fall_2011_SP, by = "StudentID", all.x = TRUE)
## next merge 2012:2016 Spring, Summer, Fall
for(i in 2012:2016) {
  df_test <- merge(x = df_test, y = get(paste0("Spring_", i, "_SP")), by = "StudentID", a
  df_test <- merge(x = df_test, y = get(paste0("Sum_", i, "_SP")), by = "StudentID", all.
  df_test <- merge(x = df_test, y = get(paste0("Fall_", i, "_SP")), by = "StudentID", all
}
## last merge Spring and Summer of 2017
df_test <- merge(x = df_test, y = Spring_2017_SP, by = "StudentID", all.x = TRUE)
df_test <- merge(x = df_test, y = Sum_2017_SP, by = "StudentID", all.x = TRUE)
```

split out data to finance, progress and statics

```
finance_train<-df_train[,1:34]
progress_train<-df_train[,-(5:66)]
static_train<-df_train[,c(1:4,35:66)]
```

```
finance_test<-df_test[,1:33]
progress_test<-df_test[,-(4:65)]
static_test<-df_test[,c(1:3,34:65)]
```

output files

```
write.csv(progress_test,file="output data/progress_test.csv",row.names = FALSE)
# write.csv(finance_test,file="output data/financial_test.csv",row.names = FALSE)
# write.csv(static_test,file="output data/static_test.csv",row.names = FALSE)
#
# write.csv(progress_train,file="output data/progress_train.csv",row.names = FALSE)
# write.csv(finance_train,file="output data/financial_train.csv",row.names = FALSE)
# write.csv(static_train,file="output data/static_train.csv",row.names = FALSE)
```

2. DATA CLEANING

2.1 Progress train data clean

```
progress_train<-read.csv("output data/progress_train.csv")
financial_train<-read.csv("output data/financial_train.csv")
static_train<-read.csv("output data/static_train.csv")

progress_train_clean<-progress_train
```

2.1.1 check if na rows

```
for (i in 0:nrow(progress_train)){
  if (sum(is.na(progress_train[i,]))>218){
    print(i)
  }
}
```

It shows that there are no rows with whole role as missing values.

We also want to check if cohort has missing values.

```
sum(is.na(progress_train_clean["cohort"]))
```

```
[1] 0
```

There are also some plot preparation.

```
progress_train_plot<-progress_train_clean
```

```
progress_train_plot$Dropout = factor(progress_train_plot$Dropout)
```

2.1.2 Complete_Dev_Math & Complete_Dev_English

In the data set, 0 means students are required to take the course but didn't, 1 means they complete it, and -1 means the information is missing. So we decided to create two other variables for Math and English to put on data: 1 means students finished, 0 means otherwise.

```
progress_train_clean$complete_DevMath=0
progress_train_clean$complete_DevEnglish=0
col_DevMath<-grep("CompleteDevMath", colnames(progress_train_clean))
col_DevEnglish<-grep("CompleteDevEnglish", colnames(progress_train_clean))

progress_train_clean[,col_DevMath][is.na(progress_train_clean[,col_DevMath])] <- "-1"
progress_train_clean[,col_DevEnglish][is.na(progress_train_clean[,col_DevEnglish])] <- "-

for (i in 1:nrow(progress_train_clean)){
  for (j in col_DevMath){
    if(progress_train_clean[i,j]==1){
      progress_train_clean[i,"complete_DevMath"]=1
    }
  }
  for (k in col_DevEnglish){
    if(progress_train_clean[i,k]==1){
      progress_train_clean[i,"complete_DevEnglish"]=1
    }
  }
}
```

2.1.3 Major

We change all the missing values in Complete1 and Complete 2 to 0, and CompleteCIP1 to -2.

```
col_complete12<-grep("Complete1|Complete2", colnames(progress_train_clean))
col_completeCIP<-grep("CompleteCIP", colnames(progress_train_clean))
progress_train_clean[,col_complete12][is.na(progress_train_clean[,col_complete12])] <- "0
progress_train_clean[,col_completeCIP][is.na(progress_train_clean[,col_completeCIP])] <-
```

We also create three more variables as `final_Completed1`, `final_Completed2`, `final_CompleteCIP1`, and `final_CompleteCIP2`.

```
###### final major
progress_train_clean$final_Complete1<-0
col_complete1<-grep("Complete1", colnames(progress_train_clean))

for (i in 1:nrow(progress_train_clean)){
```

```
  for (j in col_complete1){
    if(progress_train_clean[i,j]>0){
      progress_train_clean[i,"final_Complete1"]=progress_train_clean[i,j]
    }
  }
}

progress_train_clean$final_Complete2<-0
col_complete2<-grep("Complete2", colnames(progress_train_clean))

for (i in 1:nrow(progress_train_clean)){
  for (j in col_complete2){
    if(progress_train_clean[i,j]>0){
      progress_train_clean[i,"final_Complete2"]=progress_train_clean[i,j]
    }
  }
}

progress_train_clean$final_CompleteCIP1<-0
col_completeCIP1<-grep("CompleteCIP1", colnames(progress_train_clean))

for (i in 1:nrow(progress_train_clean)){
  for (j in col_completeCIP1){
    if(progress_train_clean[i,j]>0){
      progress_train_clean[i,"final_CompleteCIP1"]=progress_train_clean[i,j]
    }
  }
}

progress_train_clean$final_CompleteCIP2<-0
col_completeCIP2<-grep("CompleteCIP2", colnames(progress_train_clean))

for (i in 1:nrow(progress_train_clean)){
  for (j in col_completeCIP2){
    if(progress_train_clean[i,j]>0){
      progress_train_clean[i,"final_CompleteCIP2"]=progress_train_clean[i,j]
    }
  }
}
```

2.1.4 TransferIntent

We replace all the missing values in `TransferIntent` to -1.

```
col_TransferIntent<-grep("TransferIntent", colnames(progress_train_clean))
progress_train_clean[,col_TransferIntent][is.na(progress_train_clean[,col_TransferIntent]
```

2.1.5 DegreeTypeSought

2.1.5 DegreeTypeSought

We replace all the missing values in `DegreeTypeSought` to -1.

```
col_DegreeTypeSought<-grep("DegreeTypeSought", colnames(progress_train_clean))
progress_train_clean[,col_DegreeTypeSought][is.na(progress_train_clean[,col_DegreeTypeSou
```

2.1.6 GPA

We created another variable called `final_gpa` to store the students's GPA. We replace all the missing values to -1.

```
col_GPA<-grep(pattern="GPA", colnames(progress_train_clean))
progress_train_clean[,col_GPA][is.na(progress_train_clean[,col_GPA])] <- "-1"

col_CumGPA_all<-grep(pattern="CumGPA", colnames(progress_train_clean))
#add a column of final GPA
progress_train_clean$final_GPA<-0
for (i in 1:nrow(progress_train_clean)){
  for (j in col_CumGPA_all){
   if(progress_train_clean[i,j]>0){
     progress_train_clean[i,"final_GPA"]<-progress_train_clean[i,j]
   }
  }
}
```

2.1.7 Output Progress train clean files

```
# write.csv(progress_train_clean,file="output data/progress_train_clean.csv",row.names =
```

2.2 Progress test data clean

For test, we did the same thing as to train.

```
progress_test<-read.csv("output data/progress_test.csv")
financial_test<-read.csv("output data/financial_test.csv")
static_test<-read.csv("output data/static_test.csv")

progress_test_clean<-progress_test
```

2.2.1 check if na rows

```
for (i in 0:nrow(progress_test)){
  if (sum(is.na(progress_test[i,]))>218){
```

```
    print(i)
  }
}
```

It shows that there are no rows with whole role as missing values.

We also want to check if cohort has missing values.

```
sum(is.na(progress_test_clean[,"cohort"]))
```

```
[1] 0
```

Some plots.

```
progress_test_plot<-progress_test_clean
```

2.2.2 Complete_Dev_Math & Complete_Dev_English

In the data set, 0 means students are required to take the course but didn't, 1 means they complete it, and -1 means the information is missing. So we decided to create two other variables for Math and English to put on data: 1 means students finished, 0 means otherwise.

```
progress_test_clean$complete_DevMath=0
progress_test_clean$complete_DevEnglish=0
col_DevMath<-grep("CompleteDevMath", colnames(progress_test_clean))
col_DevEnglish<-grep("CompleteDevEnglish", colnames(progress_test_clean))

progress_test_clean[,col_DevMath][is.na(progress_test_clean[,col_DevMath])] <- "-1"
progress_test_clean[,col_DevEnglish][is.na(progress_test_clean[,col_DevEnglish])] <- "-1"

for (i in 1:nrow(progress_test_clean)){
  for (j in col_DevMath){
    if(progress_test_clean[i,j]==1){
      progress_test_clean[i,"complete_DevMath"]=1
    }
  }
  for (k in col_DevEnglish){
    if(progress_test_clean[i,k]==1){
      progress_test_clean[i,"complete_DevEnglish"]=1
    }
  }
}
```

2.2.3 Major

We change all the missing values in Complete1 and Complete 2 to 0, and CompleteCIP1 to -2.

```
col_complete12<-grep("Complete1|Complete2", colnames(progress_test_clean))
col_completeCIP<-grep("CompleteCIP", colnames(progress_test_clean))
progress_test_clean[,col_complete12][is.na(progress_test_clean[,col_complete12])] <- "0"
progress_test_clean[,col_completeCIP][is.na(progress_test_clean[,col_completeCIP])] <- "-
```

We also create three more variables as `final_Completed1`, `final_Completed2`, `final_CompleteCIP1`, and `final_CompleteCIP2`.

```
###### final major
progress_test_clean$final_Complete1<-0
col_complete1<-grep("Complete1", colnames(progress_test_clean))

for (i in 1:nrow(progress_test_clean)){
  for (j in col_complete1){
    if(progress_test_clean[i,j]>0){
      progress_test_clean[i,"final_Complete1"]=progress_test_clean[i,j]
    }
  }
}

progress_test_clean$final_Complete2<-0
col_complete2<-grep("Complete2", colnames(progress_test_clean))

for (i in 1:nrow(progress_test_clean)){
  for (j in col_complete2){
    if(progress_test_clean[i,j]>0){
      progress_test_clean[i,"final_Complete2"]=progress_test_clean[i,j]
    }
  }
}

progress_test_clean$final_CompleteCIP1<-0
col_completeCIP1<-grep("CompleteCIP1", colnames(progress_test_clean))

for (i in 1:nrow(progress_test_clean)){
  for (j in col_completeCIP1){
    if(progress_test_clean[i,j]>0){
      progress_test_clean[i,"final_CompleteCIP1"]=progress_test_clean[i,j]
    }
  }
}

progress_test_clean$final_CompleteCIP2<-0
col_completeCIP2<-grep("CompleteCIP2", colnames(progress_test_clean))

for (i in 1:nrow(progress_test_clean)){
  for (j in col_completeCIP2){
    if(progress_test_clean[i,j]>0){
```

```
      progress_test_clean[i,"final_CompleteCIP2"]=progress_test_clean[i,j]
    }
  }
}
```

2.2.4 TransferIntent

We replace all the missing values in `TransferIntent` to -1.

```
col_TransferIntent<-grep("TransferIntent", colnames(progress_test_clean))
progress_test_clean[,col_TransferIntent][is.na(progress_test_clean[,col_TransferIntent])]
```

2.2.5 DegreeTypeSought

We replace all the missing values in `DegreeTypeSought` to -1.

```
col_DegreeTypeSought<-grep("DegreeTypeSought", colnames(progress_test_clean))
progress_test_clean[,col_DegreeTypeSought][is.na(progress_test_clean[,col_DegreeTypeSough
```

2.2.6 GPA

We created another variable called `final_gpa` to store the students's GPA that comes from a date where its CIP is not 0. If the student has `cum_gpa` as 0, delete this student. We replace all the missing values to -999.

```
col_GPA<-grep(pattern="GPA", colnames(progress_test_clean))
progress_test_clean[,col_GPA][is.na(progress_test_clean[,col_GPA])] <- "-999"

col_CumGPA_all<-grep(pattern="CumGPA", colnames(progress_test_clean))

#add a column of final GPA
progress_test_clean$final_GPA<-0
for (i in 1:nrow(progress_test_clean)){
  for (j in col_CumGPA_all){
    if(progress_test_clean[i,j]>0){
      progress_test_clean[i,"final_GPA"]<-progress_test_clean[i,j]
    }
  }
}
```

2.2.7 Output Progress train clean files

```
# write.csv(progress_test_clean,file="output data/progress_test_clean.csv",row.names = FA
```

2.3 Finance & Static train data clean

First, we read the data.

```
finance_train <- read.csv("output data/financial_train.csv")
static_train <- read.csv("output data/static_train.csv")

finance_train[finance_train==""] <- NA
```

2.3.1 Loan, Grant

We replace all the na values in `Loan` and `Grant` to.

```
finance_train[, 11:34][is.na(finance_train[, 11:34])] <- 0
```

2.3.2 Marital.Status, Highest Grade Level, Housing.

We also replace some missing values in the front columns to “Unknown”.

```
finance_train$Marital.Status<-finance_train$Marital.Status %>% replace_na("Unknown")
finance_train$Father.s.Highest.Grade.Level<-finance_train$Father.s.Highest.Grade.Level %>
finance_train$Mother.s.Highest.Grade.Level<-finance_train$Mother.s.Highest.Grade.Level %>
finance_train$Housing<-finance_train$Housing %>% replace_na('Unknown')
```

2.3.3 Income

We replace some missing values of `income` to 0.

```
finance_train[,c("Adjusted.Gross.Income","Parent.Adjusted.Gross.Income")][is.na(finance_t
```

2.3.4 Output files.

```
static_train<-subset(static_train,select = -c(Zip,Campus,Address1,Address2,City,HSGPAWtd,

#Delete all the rows with missing State
static_train$State[is.na(static_train$State)] <- 0
static_train <- static_train[!(static_train$State=="0"),]
# static_train

# write.csv(finance_train,file="output data/finance_train_clean.csv",row.names = FALSE)
# write.csv(static_train,file="output data/static_train_clean.csv",row.names = FALSE)
```

2.4 Finance & Static test data clean

For the test set data, we almost do the same thing as train set.

First, we read the data.

```
finance <- read.csv("output data/financial_test.csv")
static <- read.csv("output data/static_test.csv")

finance[finance == ""] <- NA
```

2.4.1 Loan, Grant

We replace all the na values in `Loan` and `Grant` to.

```
finance[, 10:33][is.na(finance[, 10:33])] <- 0
```

2.4.2 Marital.Status, Highest Grade Level, Housing.

We also replace some missing values in the front columns to “Unknown”.

```
finance$Marital.Status <- finance$Marital.Status %>% replace_na("Unknown")
finance$Father.s.Highest.Grade.Level <- finance$Father.s.Highest.Grade.Level %>% replace_
finance$Mother.s.Highest.Grade.Level <- finance$Mother.s.Highest.Grade.Level %>% replace_
finance$Housing <- finance$Housing %>% replace_na('Unknown')
```

2.4.3 Income

We replace some missing values of `income` to 0.

```
finance[,c("Adjusted.Gross.Income","Parent.Adjusted.Gross.Income")][is.na(finance[,c("Adj
```

2.4.4 Output files.

```
static<-subset(static,select = -c(Zip,Campus,Address1,Address2,City,HSGPAWtd,FirstGen,Dua

#Delete all the rows with missing State
static$State[is.na(static$State)] <- 0
static <-static[!(static$State=="0"),]
# static
```

```
# write.csv(finance,file="output data/finance_test_clean.csv",row.names = FALSE)
# write.csv(static,file="output data/static_test_clean.csv",row.names = FALSE)
```

2.5 Final Merge

Finally, we merge all of our files together, which progress train merges with finance & static train, progress test merges with finance & static test.

And because `cohort` and `cohort.Term` seem to be duplicated all the time, so we want to delete them when we are merging files.

```
df_label<-read.csv("/Users/xinchangliu/Dropbox/Mac/Desktop/untitled folder 2/Student Rete
df_test<-read.csv("/Users/xinchangliu/Dropbox/Mac/Desktop/untitled folder 2/Student Reten

finance_test_clean<-read.csv("output data/finance_test_clean.csv")
finance_test_clean$StudentID <- as.numeric(finance_test_clean$StudentID)

static_test_clean<-read.csv("output data/static_test_clean.csv")
static_test_clean <- static_test_clean %>% select(-c(cohort, cohort.term))
progress_test_clean<-read.csv("output data/progress_test_clean.csv")
progress_test_clean <- progress_test_clean %>% select(-c(cohort, cohort.term))

finance_train_clean<-read.csv("output data/finance_train_clean.csv")
finance_train_clean$StudentID <- as.numeric(finance_train_clean$StudentID)
finance_train_clean <- finance_train_clean %>% select(-c(Dropout))

static_train_clean<-read.csv("output data/static_train_clean.csv")
static_train_clean <- static_train_clean %>% select(-c(Dropout,cohort, cohort.term))
progress_train_clean<-read.csv("output data/progress_train_clean.csv")
progress_train_clean <- progress_train_clean %>% select(-c(Dropout, cohort, cohort.term))


###### merge test set
df_test_clean<-merge(x=df_test,y=finance_test_clean,by="StudentID",all.x=TRUE)
df_test_clean<-merge(x=df_test_clean,y=static_test_clean,by="StudentID",all.x=TRUE)
df_test_clean<-merge(x=df_test_clean,y=progress_test_clean,by="StudentID",all.x=TRUE)

df_train_clean<-merge(x=df_label,y=finance_train_clean,by="StudentID",all.x=TRUE)
df_train_clean<-merge(x=df_train_clean,y=static_train_clean,by="StudentID",all.x=TRUE)
df_train_clean<-merge(x=df_train_clean,y=progress_train_clean,by="StudentID",all.x=TRUE)


# write.csv(df_test_clean,file="output data/df_test_clean.csv",row.names = FALSE)
# write.csv(df_train_clean,file="output data/df_train_clean.csv",row.names = FALSE)
```

3. EDA AND FEATURE ENGINEERING

3.1 Data Wrangling

3.1.1 Reading & Merging

```
test<-read.csv("/Users/xinchangliu/Dropbox/Mac/Desktop/untitled folder 2/output data/df_t
train<-read.csv("/Users/xinchangliu/Dropbox/Mac/Desktop/untitled folder 2/output data/df_
test_id <- test[,1] #test里的id
train_id <- train[,1] #train里的id
dropput_train <- train[,1:2] #train里的id和dropoutlabel
df <- bind_rows(select(train,-Dropout),test)
test_id<-as.data.frame(test_id)
names(test_id) <- "StudentID"
```

3.1.2 Variable concatenation

Variables such as major, GPA are recorded by year. These variables can not be analysis among all students because they enrolled in different years. So we first transformed them into variables unrelated to a specific year.This operation will be applied to both training and testing data sets

Major

Since there are major variables repeatedly for each semester, we decided to use the major1 and majors2 of the FIRST and LAST semester only.

```
################################      Major     ################################


####################### cohort_year ############################
for (i in 1:nrow(df)){
  df[i,"cohort_year"]<-substring((df[i,"cohort"]),1,4)
}

########   first term Major1     #########

col_Major1<-grep("Major1", colnames(df))
df[,col_Major1][is.na(df[,col_Major1])] <- "-1"
df[,"first_term_Major1"]<-0

for (i in 1:nrow(df)){
  if (df[i,"cohort.term"]==1){
    df[i,"first_term_Major1"]<-df[i,paste("Major1","Fall",df[i,"cohort_year"],sep="_")]
  }
  else if(df[i,"cohort.term"]==3){
```

```
    if(df[i,paste("Major1","Fall",df[i,"cohort_year"],sep="_")]>0){
      df[i,"first_term_Major1"]<-df[i,paste("Major1","Fall",df[i,"cohort_year"],sep="_")]
    }
    else
      df[i,"first_term_Major1"]<-df[i,paste("Major1","Spring",as.numeric(df[i,"cohort_yea
  }
}

########   first term Major2     #########
df[,"first_term_Major2"]<-0
col_Major2<-grep("Major2", colnames(df))
df[,col_Major2][is.na(df[,col_Major2])] <- "0"

for (i in 1:nrow(df)){
  if (df[i,"cohort.term"]==1){
    df[i,"first_term_Major2"]<-df[i,paste("Major2","Fall",df[i,"cohort_year"],sep="_")]
  }
  else if(df[i,"cohort.term"]==3){
    if(df[i,paste("Major2","Fall",df[i,"cohort_year"],sep="_")]>0){
      df[i,"first_term_Major2"]<-df[i,paste("Major2","Fall",df[i,"cohort_year"],sep="_")]
    }
    else
      df[i,"first_term_Major2"]<-df[i,paste("Major2","Spring",as.numeric(df[i,"cohort_yea
  }
}

########      final_majorOne     #########
col_final_majorOne<-grep("Major1", colnames(df))
df[,col_final_majorOne][is.na(df[,col_final_majorOne])] <- "-1"

df$final_majorOne <- -1

for(i in 1:nrow(df)){
  for(j in col_final_majorOne){
    if(df[i,j]>=0){
      df[i,"final_majorOne"]=df[i,j]
    }
  }
}

########      final_majorTwo     #########

col_final_majorTwo<-grep("Major2", colnames(df))
df[,col_final_majorTwo][is.na(df[,col_final_majorTwo])] <- "-1"

df$final_majorTwo <- -1

for(i in 1:nrow(df)){
  for(j in col_final_majorTwo){
    if(df[i,j]>=0){
```

```
      df[i,"final_majorIwo"]=df[i,j]
    }
  }
}

# drop orignally related variables
df <- df[ , !(grepl( "Major1_|Major2_", names(df)))]

df <- df[ , !(grepl( "Complete1_|Complete2_|CompleteCIP1_|CompleteCIP2_", names(df)))]

df <- df[ , !(grepl( "TermGPA|CumGPA", names(df)))]

df <- df[ , !(grepl( "CompleteDevMath_|CompleteDevEnglish_", names(df)))]
```

TransferIntent and DegreeTypeSought

We also want to extract the final TransferIntent and DegreeTypeSought of the last semeste for each student as well as whether they once have TransferIntent.

```
########    final_transferIt ###################

col_final_transferIt<-grep("TransferIntent", colnames(df))
df[,col_final_transferIt][is.na(df[,col_final_transferIt])] <- "-1"
df$final_transferIt <- -1

for (i in 1:nrow(df)){
  for (j in col_final_transferIt){
    if(df[i,j]>=0){
      df[i,"final_transferIt"]=df[i,j]
    }
  }
}
###################  final_degreeSought  ##############################

col_final_degreeSought<-grep("DegreeTypeSought", colnames(df))
df[,col_final_degreeSought][is.na(df[,col_final_degreeSought])] <- "-1"
df$final_degreeSought <- -1
for (i in 1:nrow(df)){
  for (j in col_final_degreeSought){
    if(df[i,j]>=1){
      df[i,"final_degreeSought"]=df[i,j]
    }
  }
}

###################### once_TransferIntent  ##############################

col_TransferIntent<-grep("TransferIntent", colnames(df))
df[,col_TransferIntent][is.na(df[,col_TransferIntent])] <- "-2"
```

```
df[,"once_TransferIntent"]=0
for (i in 1:nrow(df)){
  for (j in col_TransferIntent){
    if(df[i,j]>0){
      df[i,"once_TransferIntent"]=1
    }
  }
}



# drop orignally related variables
df <- df[ , !(grepl( "TransferIntent_|DegreeTypeSought_", names(df)))]
```

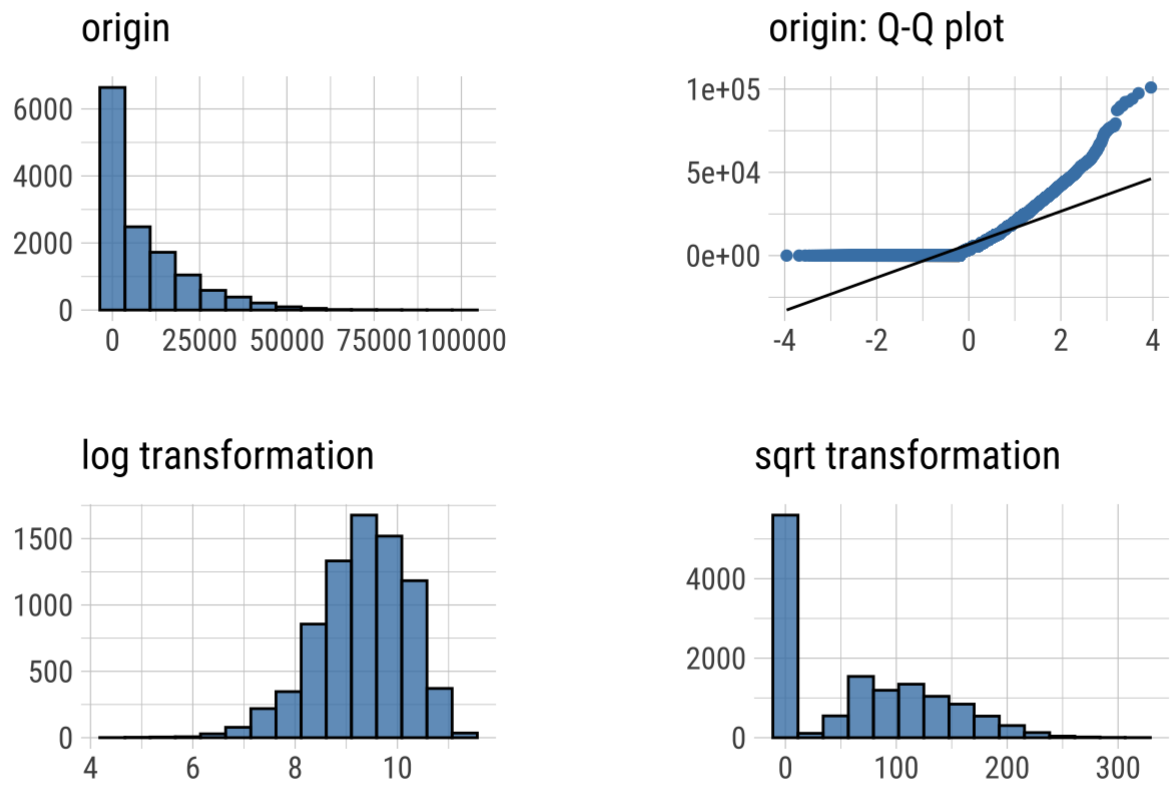
Loan|Work.Study|Grant|Scholarship

We have extracted the total amount of each subsidy obtained by students during their college years.

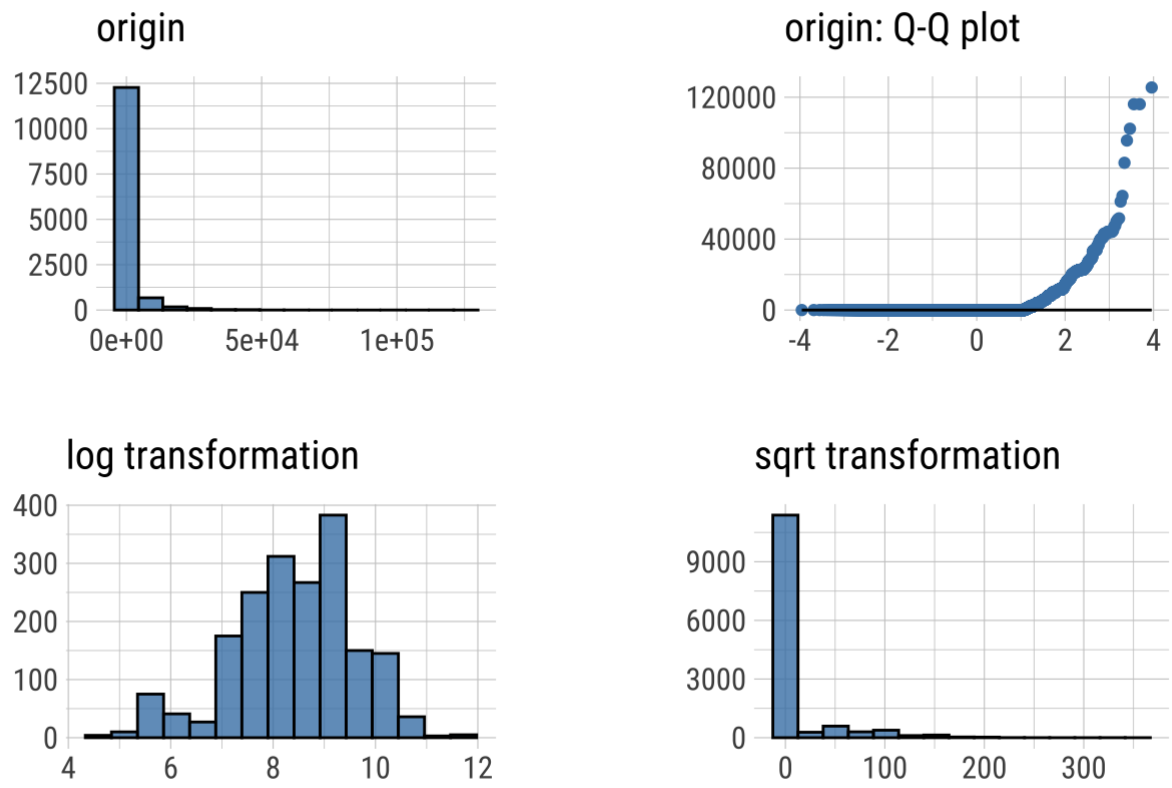
```
######      Loan|Work.Study|Grant|Scholarship      ############

for (i in 1:nrow(df)){
  df$total_Loan[i]<-sum(df[i,grep("\\.Loan", colnames(df))])
}

for (i in 1:nrow(df)){
  df$total_Scholarship[i]<-sum(df[i,grep("\\.Scholarship", colnames(df))])
}

for (i in 1:nrow(df)){
  df$total_Work_Study[i]<-sum(df[i,grep("\\.Work\\.Study", colnames(df))])
}

for (i in 1:nrow(df)){
  df$total_Grant[i]<-sum(df[i,grep("\\.Grant", colnames(df))])
}

df <- df[ , !(grepl( "\\.Loan|\\.Work.Study|\\.Grant|\\.Scholarship", names(df)))]
```

We now look at the their distribution: Based on the plots, we can see that after log transformation, the distribution of is less skewed. As a result, we need to do the log transformation on those four variables.

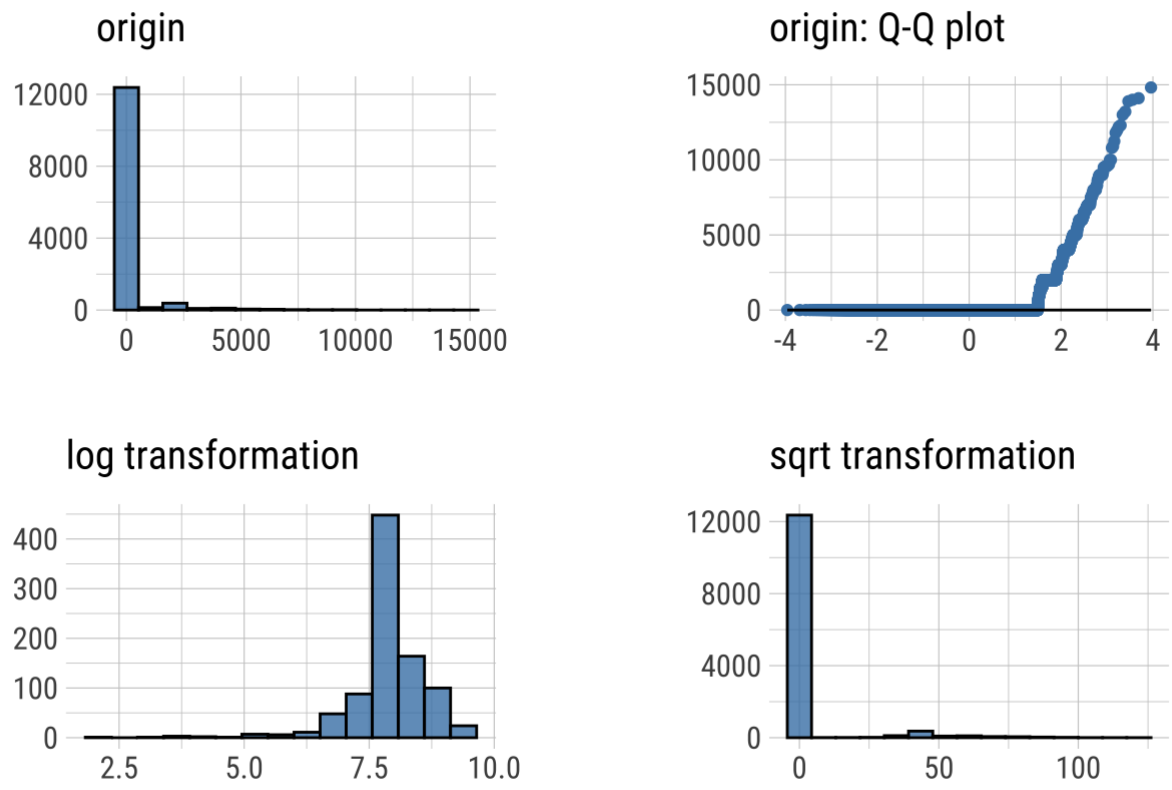
```
plot_normality(df,total_Loan)
```



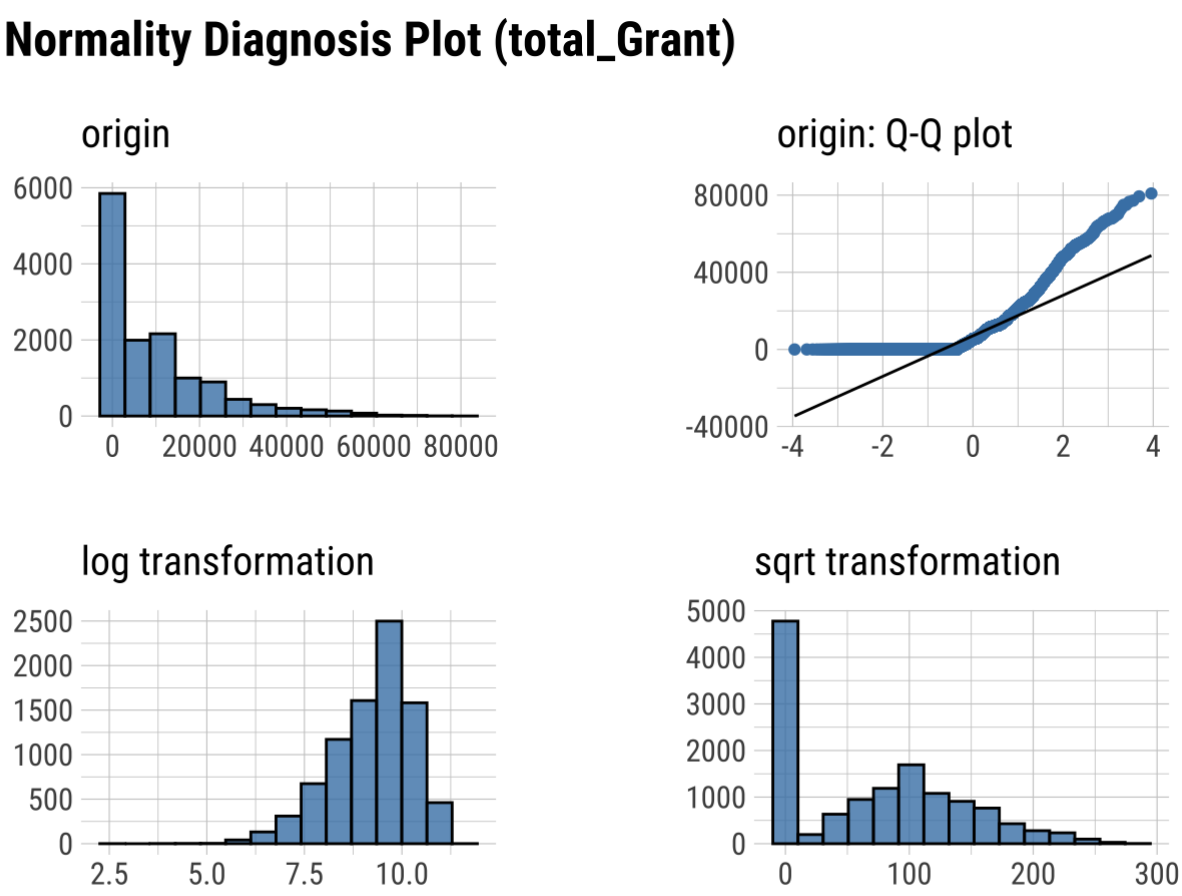

```
plot_normality(df,total_Scholarship)
```




```
plot_normality(df,total_Work_Study)
```




```
plot_normality(df,total_Grant)
```




```
df$total_Loan <- log10(df$total_Loan)
df$total_Loan[df$total_Loan=="-Inf"] <- 0
df$total_Scholarship <- log10(df$total_Scholarship)
df$total_Scholarship[df$total_Scholarship=="-Inf"] <- 0
df$total_Work_Study <- log10(df$total_Work_Study)
df$total_Work_Study[df$total_Work_Study=="-Inf"] <- 0
df$total_Grant <- log10(df$total_Grant)
df$total_Grant[df$total_Grant=="-Inf"] <- 0
```

Race

Instead of using seven separate race indicators (which are perfectly correlated), we decided to add a new variable called race to indicate a student's race directly and drop seven indicators.

```
################################      Race      ##############################
df$race <- ifelse(df$Hispanic==1,"Hispanic", ifelse(df$AmericanIndian==1,"AmericanIndian"
df[which(df$Asian==-1),"race"] <- "Unknown"

# drop original seven race indicators
df <- df %>% select(-c(Hispanic,AmericanIndian,Asian,Black,NativeHawaiian,White,TwoOrMore
```

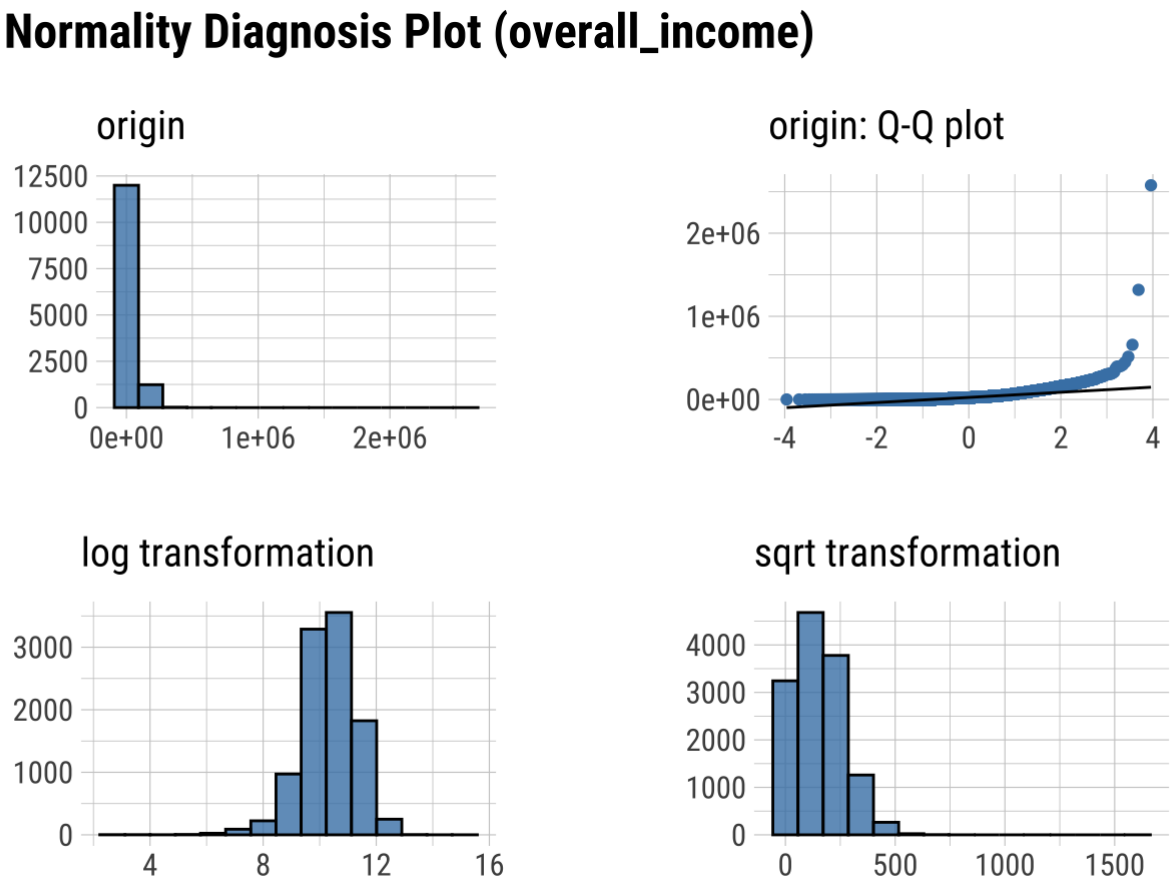
Income

We found a negative value in income and we assume that minus sign is due to a mistake. So we first took the absolute value. And according to the table, each income variable has half of the values = 0, we decide to add these two together as an overall income.

```
###### abs
df$Adjusted.Gross.Income<-abs(df$Adjusted.Gross.Income)
df$Parent.Adjusted.Gross.Income<-abs(df$Parent.Adjusted.Gross.Income)

###### overall_income  #############
df$overall_income <- apply(select(df,c(Adjusted.Gross.Income,Parent.Adjusted.Gross.Income
df$overall_income <- abs(df$overall_income)
#df <- df %>% select(-c(Adjusted.Gross.Income,Parent.Adjusted.Gross.Income))
```

Same as financial aid, we look at the distribution: Based on the plots, we can see that after log transformation, the distribution of is less skewed. As a result, we need to do the log transformation.

```
plot_normality(df,overall_income)
```



```
df$overall_income <- log10(df$overall_income)
df$overall_income[df$overall_income=="-Inf"] <- 0
```

3.1.3 Create variables

In this session, we generated some variables based on our perceptual intuition

Generate enrollment age

By combining the column of Birth year and cohort term, we can get the age when students start their cohort term in the university. Different age people might have different cerebral function developed level, or deteriorated condition. Especially the students provided comes from a wide range of age group, from 15 to over 60 when they have their first cohort term. But the difference will be more distinguished in comparing different age group than comparing similar ages, so students are divided as different age group by 10 year difference, which is also the difference of generation.

```
#########################generate enrollment age ############################
del<-which(is.na(df$"BirthYear"),arr.ind = TRUE)
df<-df[-del,]
for (i in 1:nrow(df)){
  df[i,"enrolled_age"]<-as.numeric(df[i,"cohort_year"])-as.numeric(df[i,"BirthYear"])
  if(as.numeric(df[i,"cohort_year"])-as.numeric(df[i,"BirthYear"])>100){
    print(i)
  }
}
```

3.1.4 Floor and factors

```
##### floor major

for (i in c("final_majorOne","final_majorTwo","first_term_Major1","first_term_Major2")){
  df[,i] <- as.numeric(df[,i])
  df[,i] <- floor(df[,i])
}

##### factor

# # code categorical variables into factors
# df$cohort <- factor(df$cohort)
# df$cohort.term <- factor(df$cohort.term, levels=c(1:7), labels=c("Term 1","Term 2","Ter
# df$Marital.Status <- factor(df$Marital.Status)
# df$Father.s.Highest.Grade.Level <- factor(df$Father.s.Highest.Grade.Level)
# df$Mother.s.Highest.Grade.Level <- factor(df$Mother.s.Highest.Grade.Level)
# df$Housing <- factor(df$Housing)
# df$Gender <- factor(df$Gender, levels=c(1,2,3,-1), labels=c("Male","Female","Other","Mi
# df$HSDip <- factor(df$HSDip, levels=c(0,1,2,3,4,-1), labels=c("None","HighSchoolDiploma
# df$EnrollmentStatus <- factor(df$EnrollmentStatus,levels=c(1,2,-1),labels=c("EnteringFr
# df$HighDeg <- factor(df$HighDeg,levels=c(0,1,2,3,4,5,-1),labels=c("None","CertificateUn
```

```
# df$MathPlacement <- factor(df$MathPlacement,levels=c(0,1,-1), labels=c("ready","notread
# df$EngPlacement <- factor(df$EngPlacement,levels=c(0,1,-1), labels=c("ready","notready"
# df$GatewayEnglishStatus <- factor(df$GatewayEnglishStatus,levels=c(0,1,-1), labels=c("n
# df$GatewayMathStatus <- factor(df$GatewayMathStatus,levels=c(0,1,-1), labels=c("notrequ
# df$complete_DevEnglish <- factor(df$complete_DevEnglish,levels=c(0,1),labels=c("notcomp
# df$complete_DevMath <- factor(df$complete_DevMath,levels=c(0,1),labels=c("notcomplete",
# df$race <- factor(df$race)
# df$final_degreeSought <- factor(df$final_degreeSought, levels=c(1,2,3,4,5,6,-1), labels
# df$BirthMonth <- factor(df$BirthMonth)
# df$State <- factor(df$State)
#
#
# df$final_MajorOne <- factor(df$final_majorOne)
# df$final_MajorTwo <- factor(df$final_majorTwo)
# df$final_first_term_Major1 <- factor(df$first_term_Major1)
# df$final_first_term_Major2 <- factor(df$first_term_Major2)
#
#
# df$final_Complete1 <- factor(df$final_Complete1)
# df$final_Complete2 <- factor(df$final_Complete2)
# df$final_CompleteCIP1 <- factor(df$final_CompleteCIP1)
# df$final_CompleteCIP2 <- factor(df$final_CompleteCIP2)


#df$Dropout = factor(df$Dropout,levels = c(0,1),labels = c("Grad", "dropout"))
```

3.1.5 Check NA and DOESN'T APPLY proporation

Examine which variables have too many missing values (more than 50%)or doesn't apply condition according to the code book.

```
observations <- nrow(df)

########################  Continuous Variables  ########################
for (i in c("HSGPAUnwtd","NumColCredAttemptTransfer","NumColCredAcceptTransfer","CumLoanA
  if (sum(df[,i]==-1)/observations>0.5){
  print(paste("na/unknown ratio of",i,sum(df[,i]==-1)/observations))
  }
}
```

```
[1] "na/unknown ratio of HSGPAUnwtd 0.7026395173454"
[1] "na/unknown ratio of CumLoanAtEntry 0.588838612368024"
```

```
for (i in c("Adjusted.Gross.Income","Parent.Adjusted.Gross.Income","overall_income")){
  if (sum(df[,i]==-1)/observations>0.5){
  print(paste("na/unknown ratio of",i,sum(df[,i]==0)/observations))
```

```
  }
}


######################  Discrete variable  ######################
observations <- nrow(df)
for (i in c("Marital.Status","Father.s.Highest.Grade.Level","Mother.s.Highest.Grade.Level
  if (sum(df[,i]=="Unknown")/observations>0.5){
  print(paste("na/unknown ratio of",i,sum(df[,i]=="Unknown")/observations))
  }
}

for (i in c("HSDip","EnrollmentStatus","MathPlacement","EngPlacement","GatewayMathStatus"
  if (sum(df[,i]=="missing")/observations>0.5){
  print(paste("na/unknown ratio of",i,sum(df[,i]=="missing")/observations))
  }
}

for (i in c("HSDipYr","HighDeg","complete_DevMath","complete_DevEnglish","final_degreeSou
  if (sum(df[,i]==-1)/observations>0.5){
  print(paste("na/unknown ratio of",i,sum(df[,i]==-1)/observations))
  }
}
```

```
[1] "na/unknown ratio of HSDipYr 0.722473604826546"
[1] "na/unknown ratio of final_transferIt 1"
[1] "na/unknown ratio of first_term_Major2 0.983634992458522"
```

```
################## drop  ##################
df <- df %>% select(-HSGPAUnwtd)
df <- df %>% select(-CumLoanAtEntry)
df <- df %>% select(-final_transferIt)
```

For Continuous variables have too many missing values, we decide to drop them:

1. Over 58% of students have missing variable CumLoanAtEntry value.
2. Over 70% of students have missing variable HSGPAUnwtd value.

For Discrete variables have too many missing values, we would keep them temporarily until we test whether those NA has potential meaning (for example, whether the dropout rate of students recorded as na in some variables is significantly different from that of students with values)（在train中检测）

1. According to our calculations, over 72% of students have missing variable HSDipYr value.

2. Over 98% of students have missing variable first_term_major2 value.

Also, variable final_transferIt only contains -1, which refers to missing values, so we drop this feature.

3.1.6 Check variables of near zero variance

If a variable has very little change or variation, it's like a constant and not useful for prediction so we would like to drop them.

```
########## variables of near zero variance
colnames(df)[nearZeroVar(df)]
```

```
 [1] "State"               "HSDip"               "final_Complete2"
 [4] "final_CompleteCIP1"  "final_CompleteCIP2"  "first_term_Major2"
 [7] "final_majorTwo"      "final_degreeSought"  "once_TransferIntent"
[10] "total_Scholarship"   "total_Work_Study"
```

```
########## drop them
df <- select(df,-colnames(df)[nearZeroVar(df)][1:9])
```

11 variables with low variances are：

"State", "HSDip", "final_Complete2", "final_CompleteCIP1", "final_CompleteCIP2", "first_term_Major2","final_majorTwo" , "final_degreeSought", "once_TransferIntent", "total_Scholarship", "total_Work_Study"

3.1.7 Extract train data set for EDA

```
train_EDA <- left_join(dropput_train, df, by="StudentID")
train_EDA$Dropout <- factor(train_EDA$Dropout)
```

3.2 Exploratory Data Analysis & Feature Engineering

3.2.1 Interval & Ratio Level Measures（Continuous Variables）:

variables overview：

	Static	Progress
Adjusted.Gross.Income	BirthYear,	final_GPA
Parent.Adjusted.Gross.Income	HSDipYr	valid_transfer
overall_income	NumColCredAcceptTransfer	
Scholarship	NumColCredAttemptTransfer	
Work.Study	enrolled_age	
Grant	RegistrationDate	

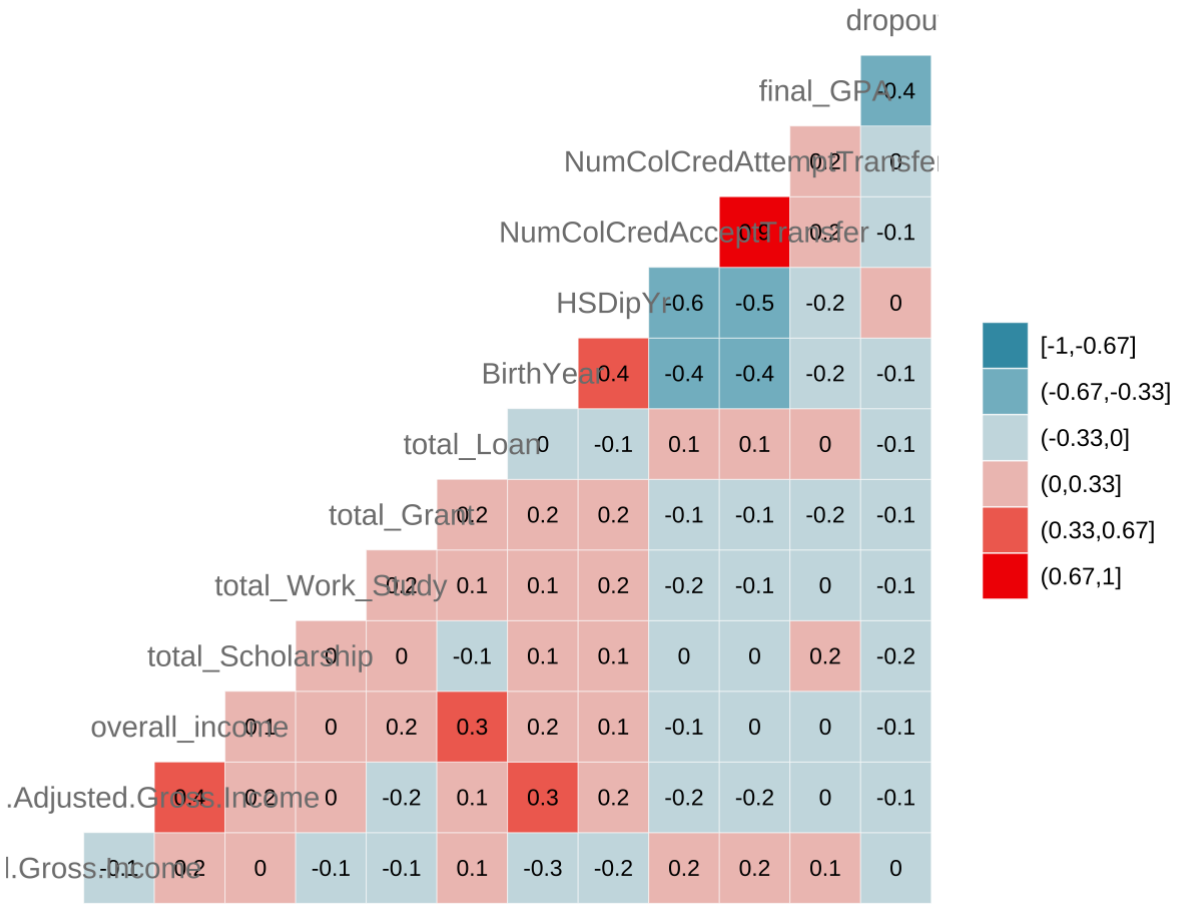
Static	Progress

Loan

Correlation test

Firstly, we conducted a correlation test

```
ggcorr(train_EDA[,c("Adjusted.Gross.Income","Parent.Adjusted.Gross.Income","overall_incom
    nbreaks = 6,
    label = TRUE,
    label_size = 3,
    color = "grey50")
```



Then we look at each variables:

Income

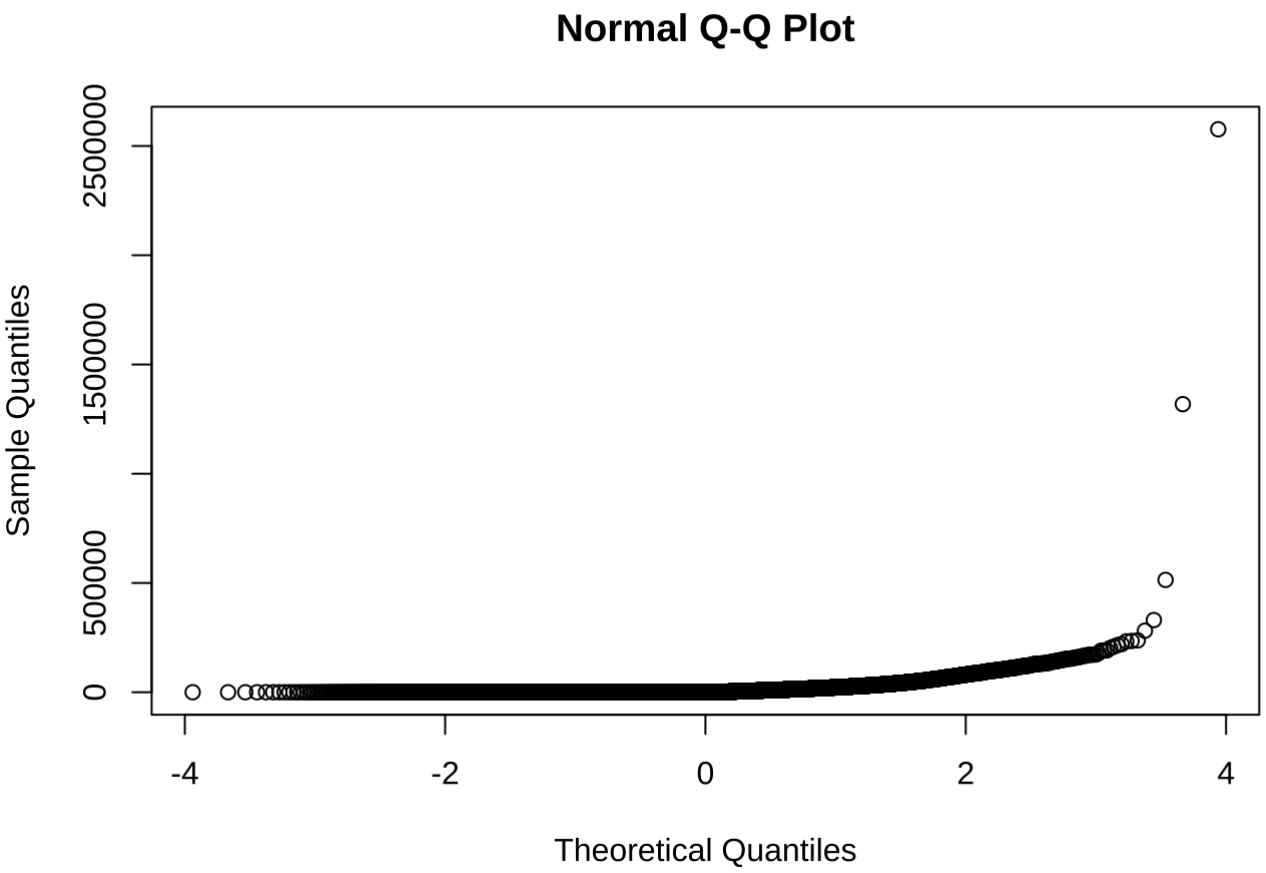
```
######  Adjusted.Gross.Income, Parent.Adjusted.Gross.Income, overall_income

###normal distribution test
```

```
ks.test(scale(train_EDA$Adjusted.Gross.Income),"pnorm")
```

```
Warning in ks.test.default(scale(train_EDA$Adjusted.Gross.Income), "pnorm"):
ties should not be present for the Kolmogorov-Smirnov test
```

```
    Asymptotic one-sample Kolmogorov-Smirnov test

data:  scale(train_EDA$Adjusted.Gross.Income)
D = 0.37099, p-value < 2.2e-16
alternative hypothesis: two-sided
```

```
qqnorm(train_EDA$Adjusted.Gross.Income)
```



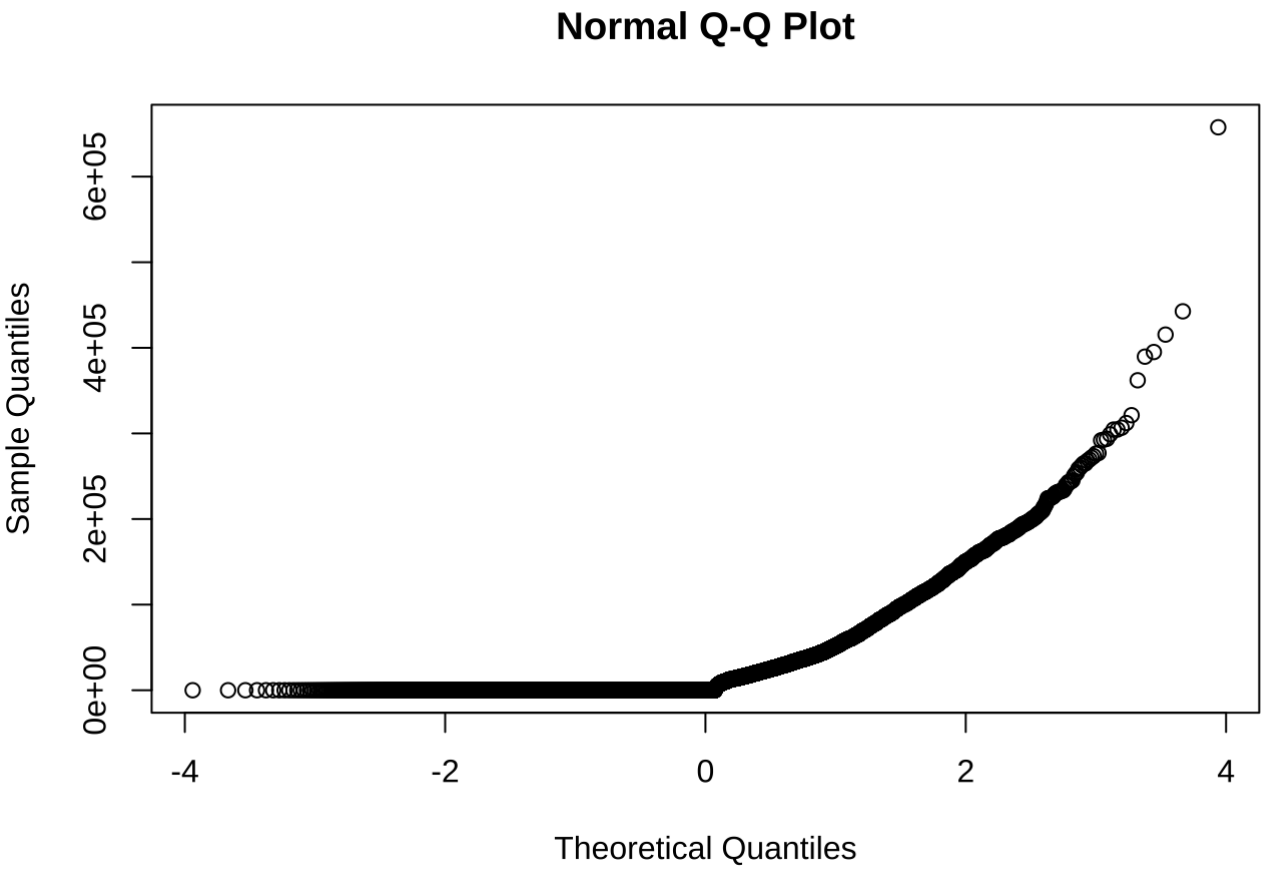

```
ks.test(scale(train_EDA$Parent.Adjusted.Gross.Income),"pnorm")
```

```
Warning in ks.test.default(scale(train_EDA$Parent.Adjusted.Gross.Income), : ties
should not be present for the Kolmogorov-Smirnov test
```

```
    Asymptotic one-sample Kolmogorov-Smirnov test

data:  scale(train_EDA$Parent.Adjusted.Gross.Income)
```

```
D = 0.27923, p-value < 2.2e-16
alternative hypothesis: two-sided
```

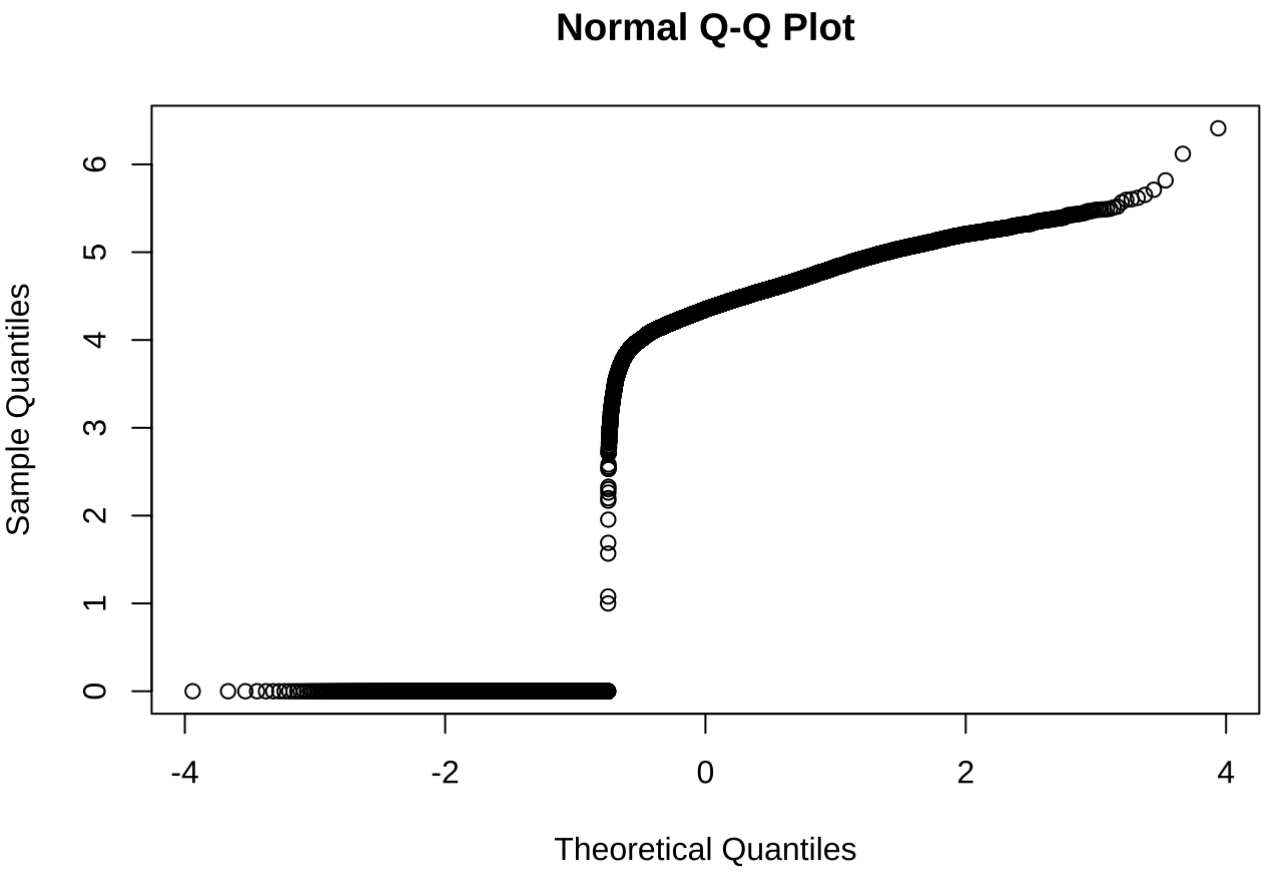
```
qqnorm(train_EDA$Parent.Adjusted.Gross.Income)
```




```
ks.test(scale(train_EDA$overall_income),"pnorm")
```

```
Warning in ks.test.default(scale(train_EDA$overall_income), "pnorm"): ties
should not be present for the Kolmogorov-Smirnov test


    Asymptotic one-sample Kolmogorov-Smirnov test

data:  scale(train_EDA$overall_income)
D = 0.30981, p-value < 2.2e-16
alternative hypothesis: two-sided
```

```
qqnorm(train_EDA$overall_income)
```




```
####  Mann-Whitney U test
wilcox.test(train_EDA$Parent.Adjusted.Gross.Income~train_EDA$Dropout)
```

```
    Wilcoxon rank sum test with continuity correction

data:  train_EDA$Parent.Adjusted.Gross.Income by train_EDA$Dropout
W = 19269264, p-value < 2.2e-16
alternative hypothesis: true location shift is not equal to 0
```

```
wilcox.test(train_EDA$Adjusted.Gross.Income~train_EDA$Dropout)
```

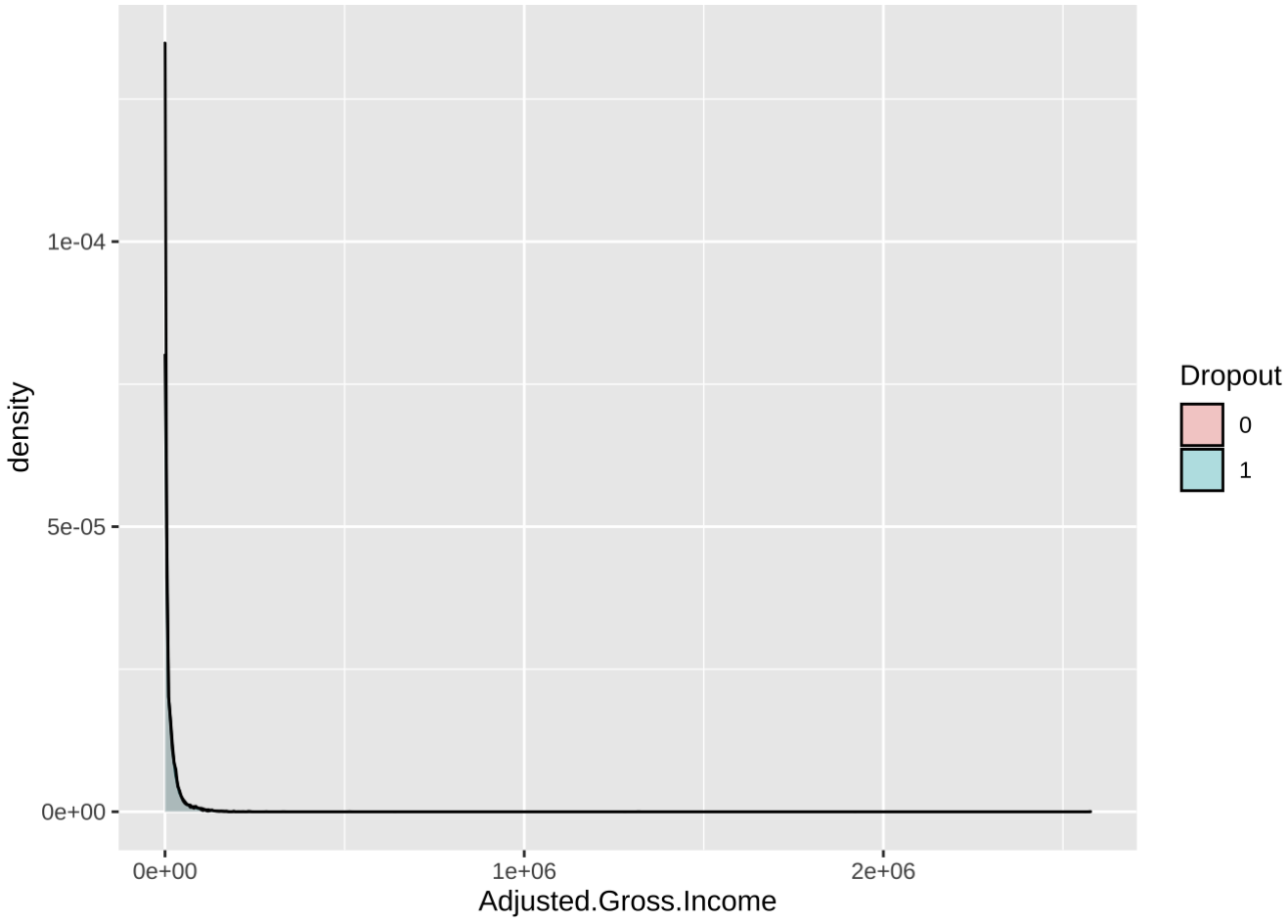
```
    Wilcoxon rank sum test with continuity correction

data:  train_EDA$Adjusted.Gross.Income by train_EDA$Dropout
W = 19069780, p-value = 1.022e-12
alternative hypothesis: true location shift is not equal to 0
```

```
wilcox.test(train_EDA$overall_income~train_EDA$Dropout)
```

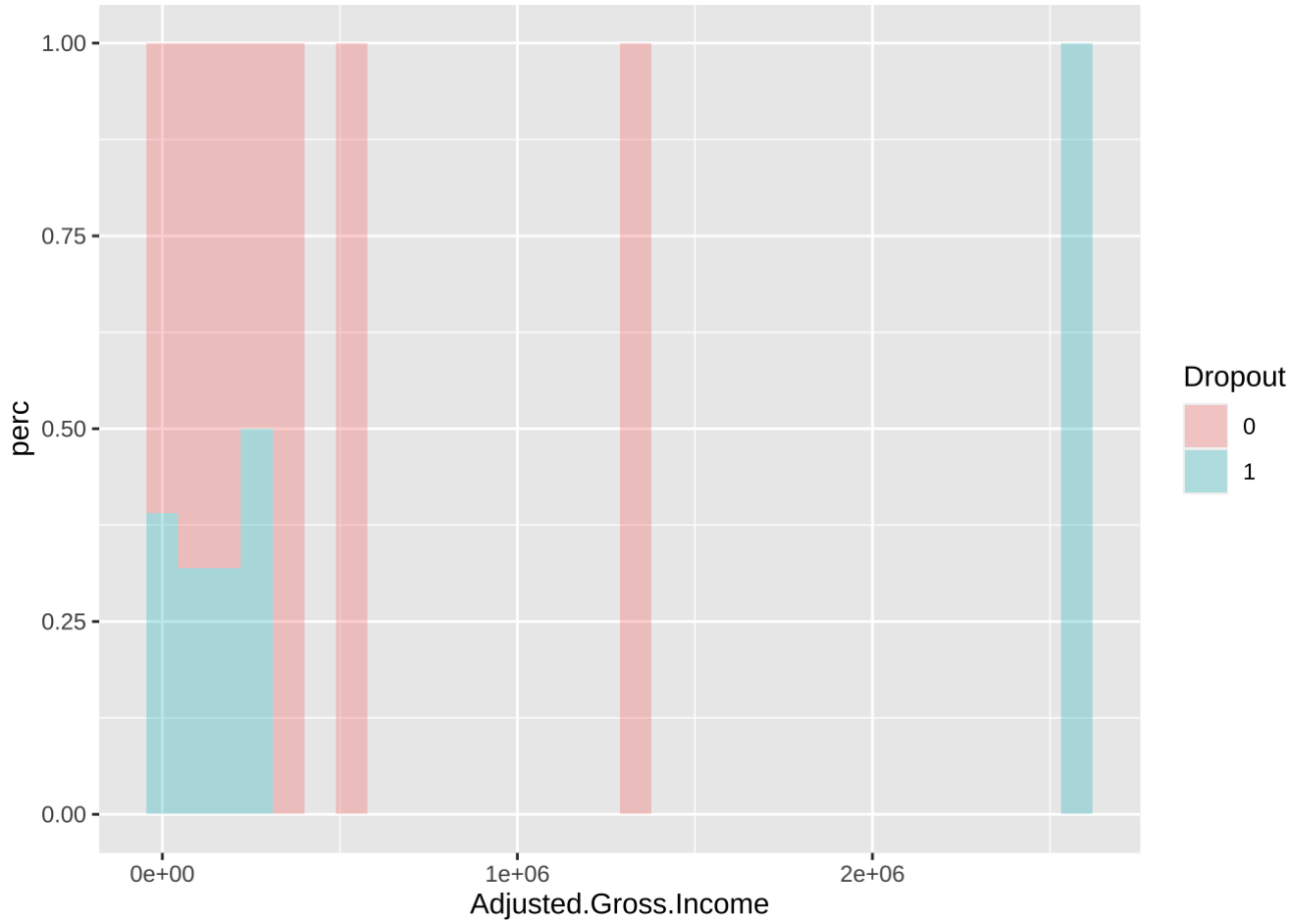
```
    Wilcoxon rank sum test with continuity correction

data:  train_EDA$overall_income by train_EDA$Dropout
W = 20173982, p-value < 2.2e-16
alternative hypothesis: true location shift is not equal to 0
```

```
for(i in c("Adjusted.Gross.Income","Parent.Adjusted.Gross.Income","overall_income")){
print(ggplot(train_EDA, aes(x = get(i), fill = Dropout)) +xlab(i)+geom_density(alpha = 0.
print(ggplot(data = train_EDA, mapping = aes(x = get(i), fill =Dropout)) +xlab(i)+ ylab("
}
```

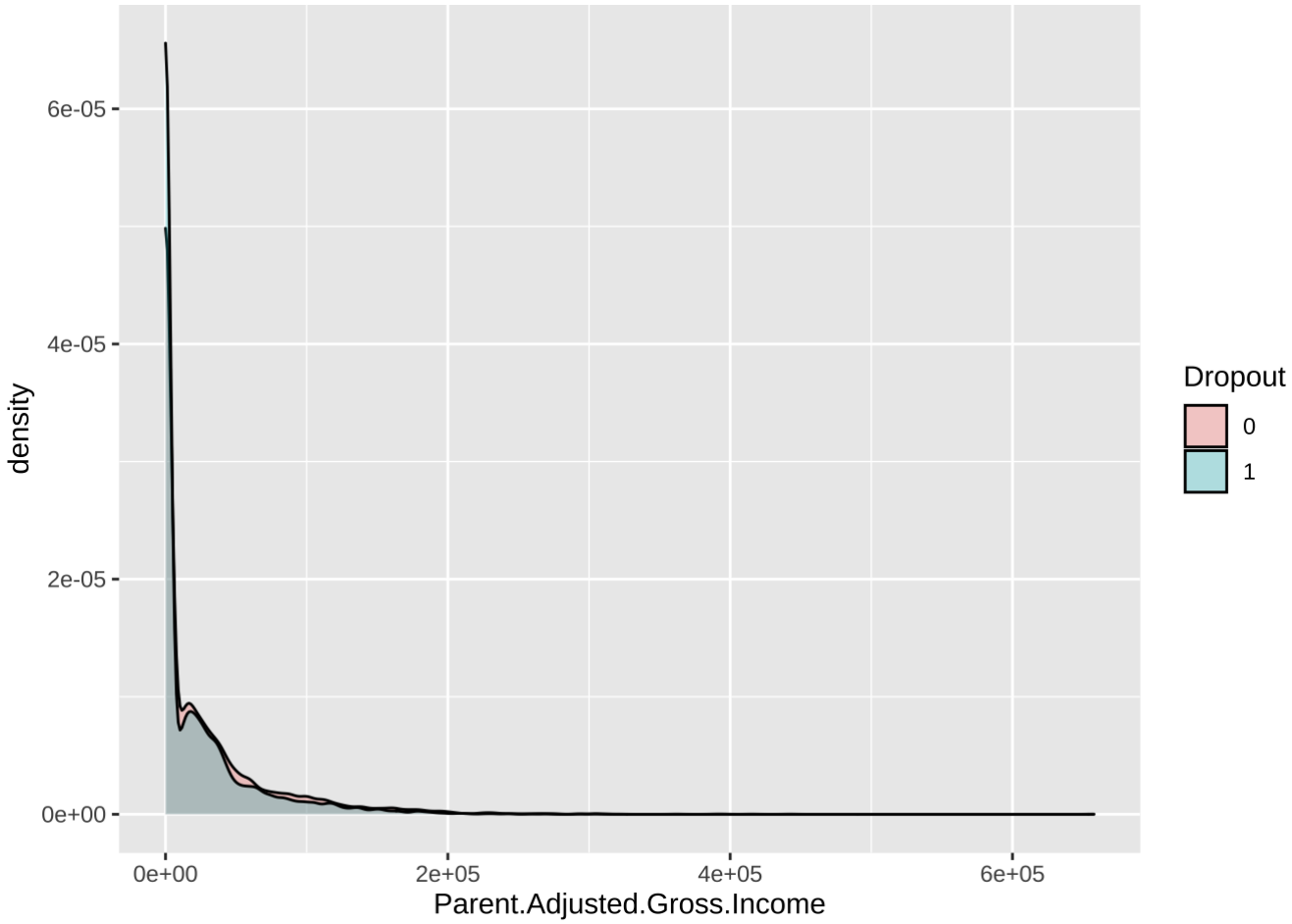
```
Warning: Removed 1 rows containing non-finite values (stat_density).
```



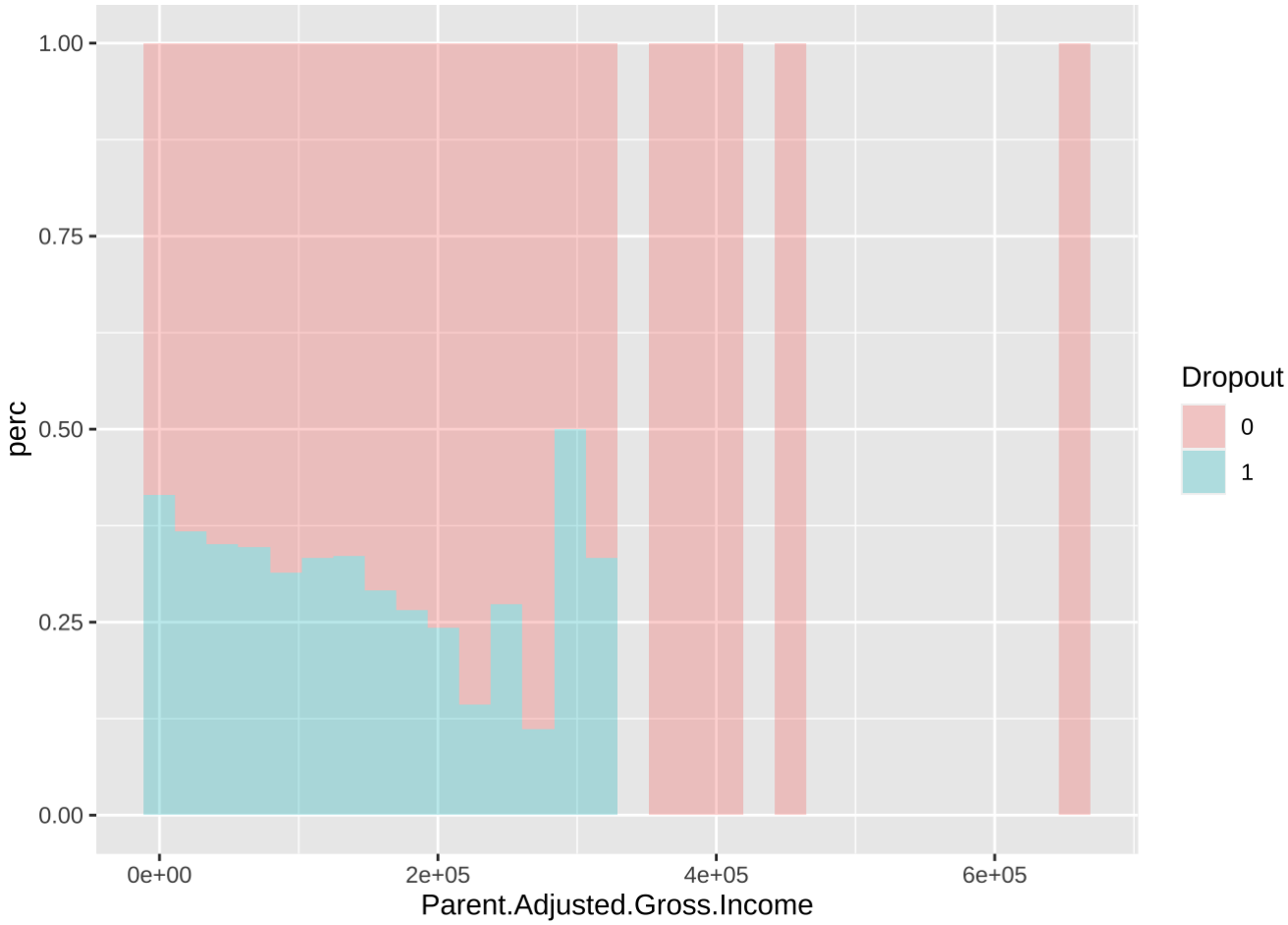

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

Warning: Removed 1 rows containing non-finite values (stat_bin).

Warning: Removed 44 rows containing missing values (geom_bar).
```



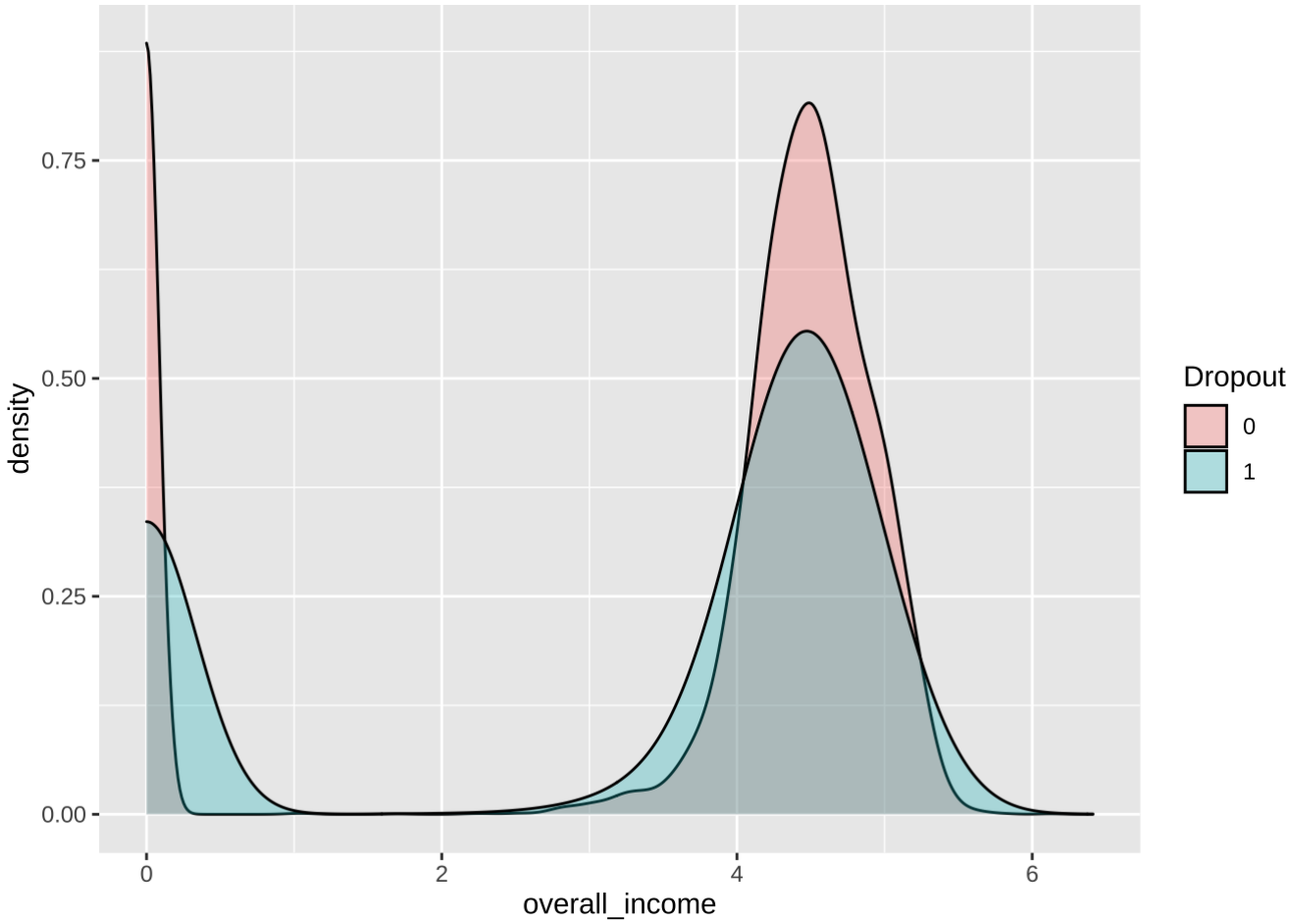

```
Warning: Removed 1 rows containing non-finite values (stat_density).
```



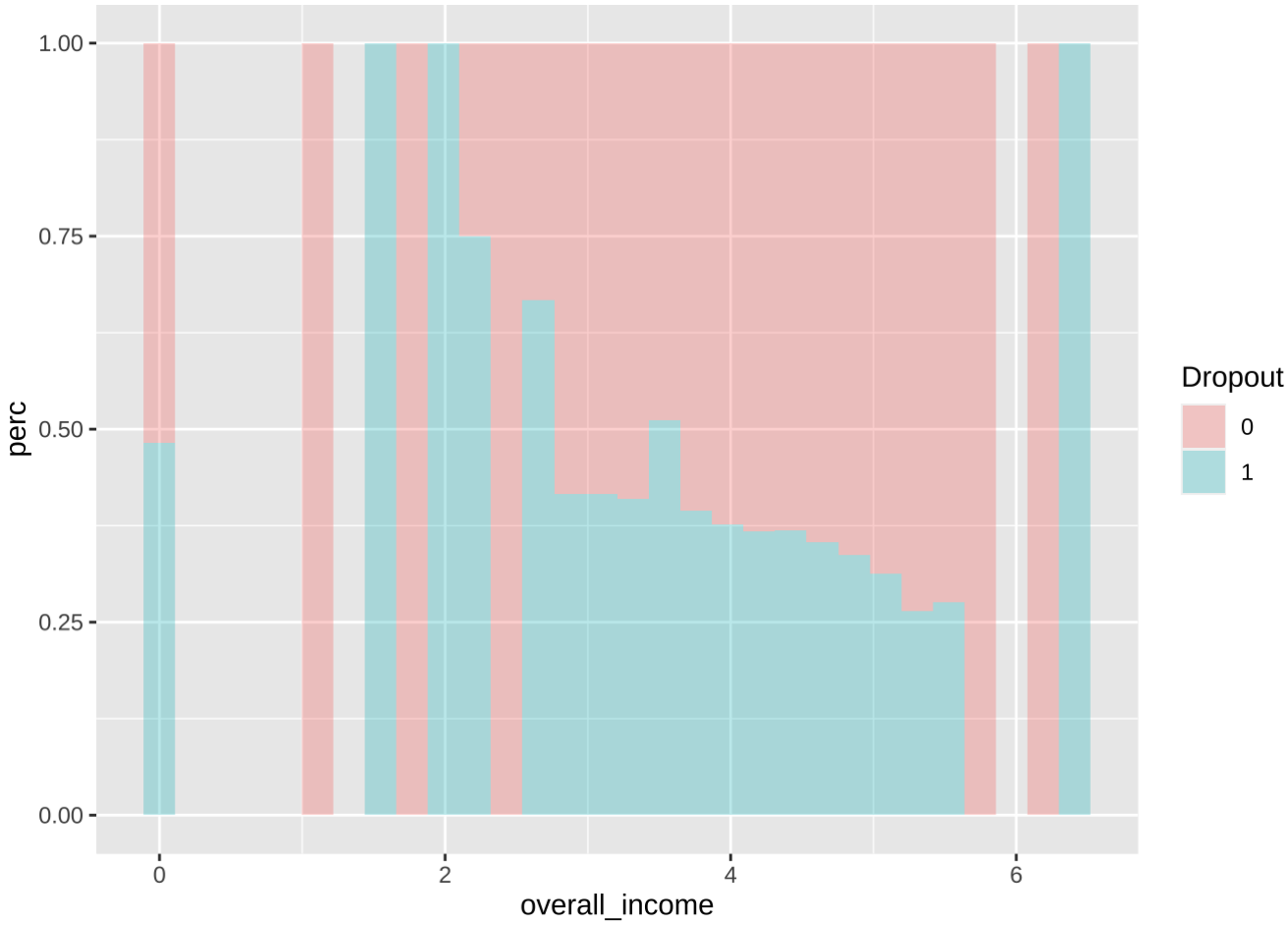

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

Warning: Removed 1 rows containing non-finite values (stat_bin).

Warning: Removed 20 rows containing missing values (geom_bar).
```




```
Warning: Removed 1 rows containing non-finite values (stat_density).
```




```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

Warning: Removed 1 rows containing non-finite values (stat_bin).

Warning: Removed 12 rows containing missing values (geom_bar).
```



Analysis and Main Results:

None of these three variables are normally distributed, so Wilcoxon rank sum test was conducted and results show that all these three variables are significantly correlated with dropout.

So all these three variables are left and could be further selected (Parent.Adjusted.Gross.Income and Adjusted.Gross.Income, or overall_income) in the model

Involved variables:

- Adjusted.Gross.Income & Parent.Adjusted.Gross.Income | overall_income (newly generated)

Financial aids:

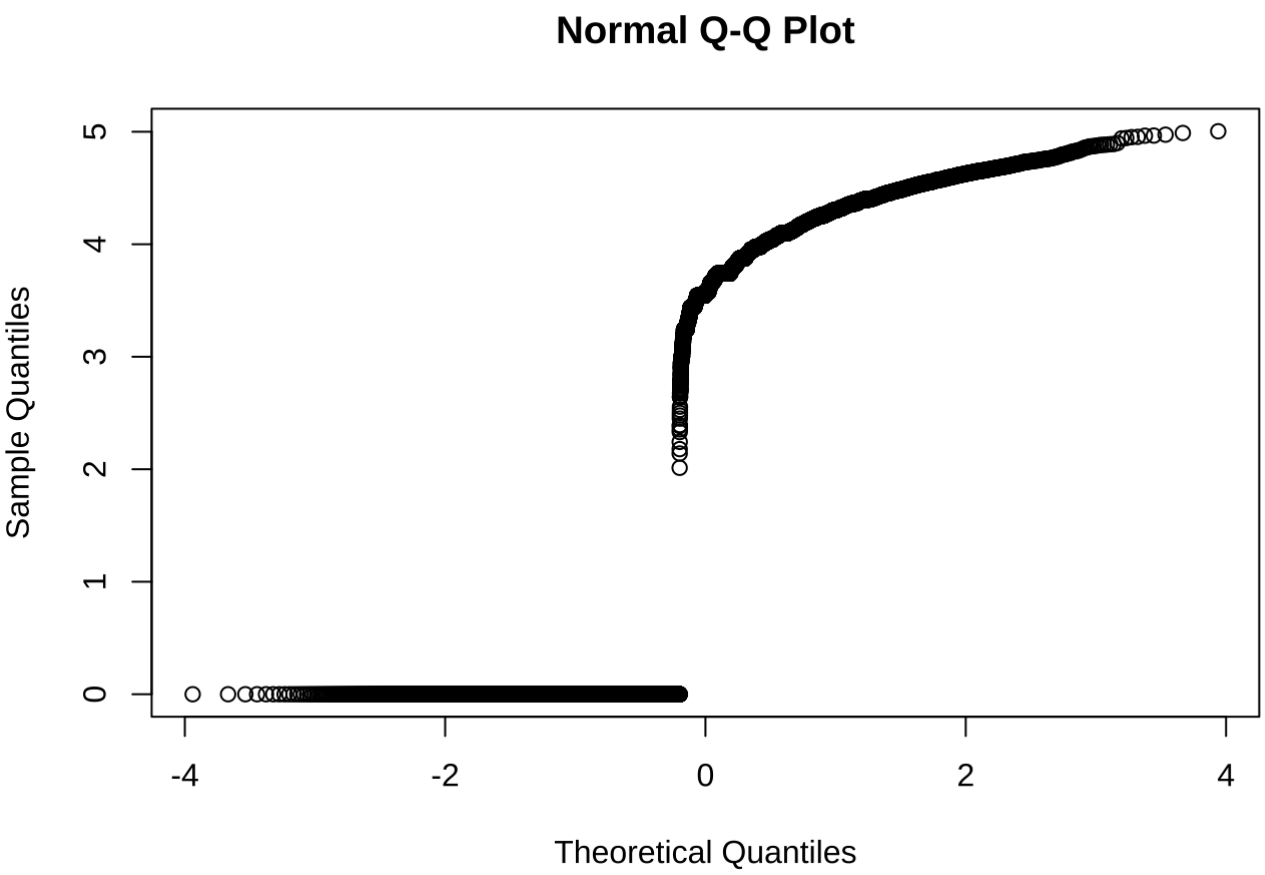
```
####################Loan|Work.Study|Grant|Scholarship ####################

############  normal distribution test
ks.test(scale(train_EDA$total_Loan),"pnorm")
```

```
Warning in ks.test.default(scale(train_EDA$total_Loan), "pnorm"): ties should
not be present for the Kolmogorov-Smirnov test
```

```
    Asymptotic one-sample Kolmogorov-Smirnov test
```

```
data:  scale(train_EDA$total_Loan)
D = 0.29813, p-value < 2.2e-16
alternative hypothesis: two-sided
```

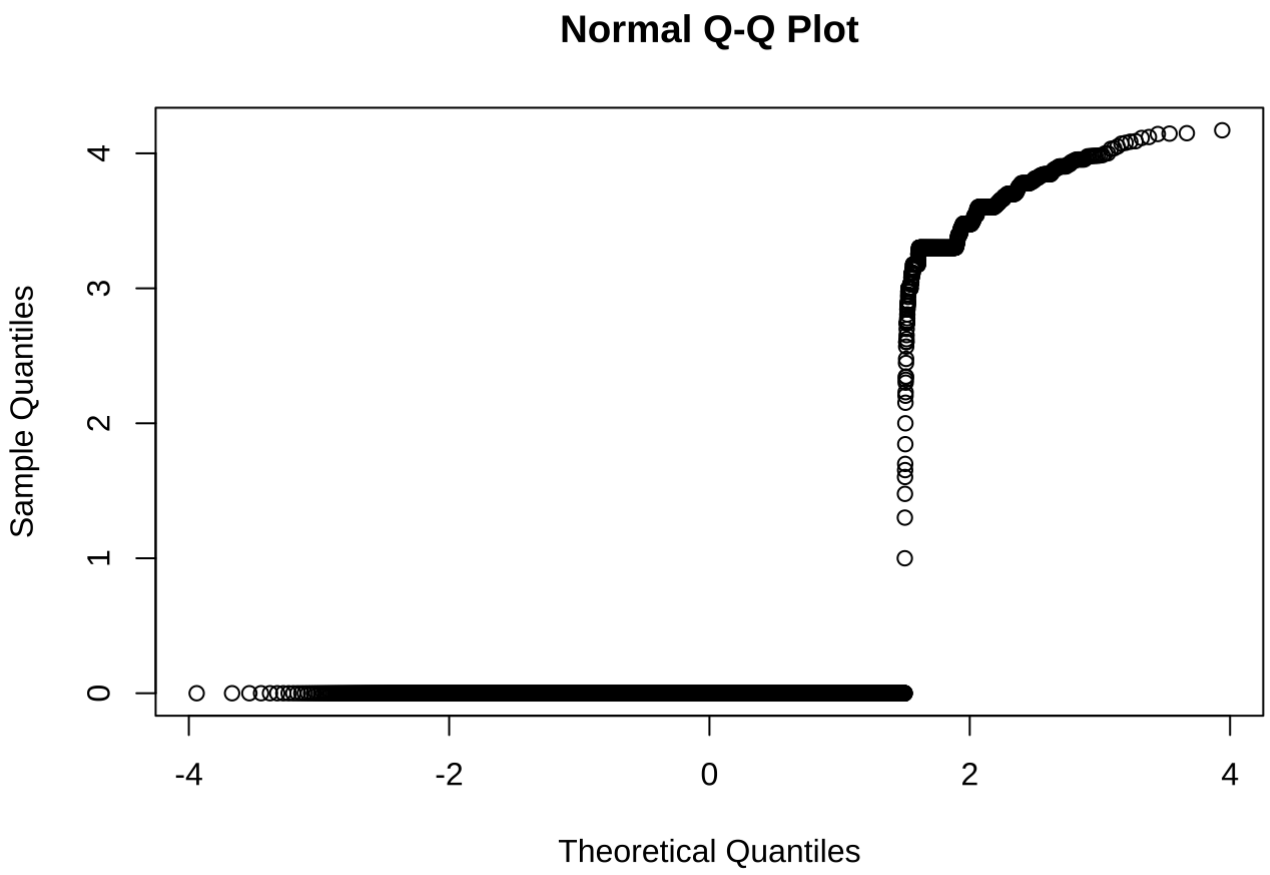
```
qqnorm(train_EDA$total_Loan)
```




```
ks.test(scale(train_EDA$total_Work_Study),"pnorm")
```

```
Warning in ks.test.default(scale(train_EDA$total_Work_Study), "pnorm"): ties
should not be present for the Kolmogorov-Smirnov test


    Asymptotic one-sample Kolmogorov-Smirnov test

data:  scale(train_EDA$total_Work_Study)
D = 0.53812, p-value < 2.2e-16
alternative hypothesis: two-sided
```

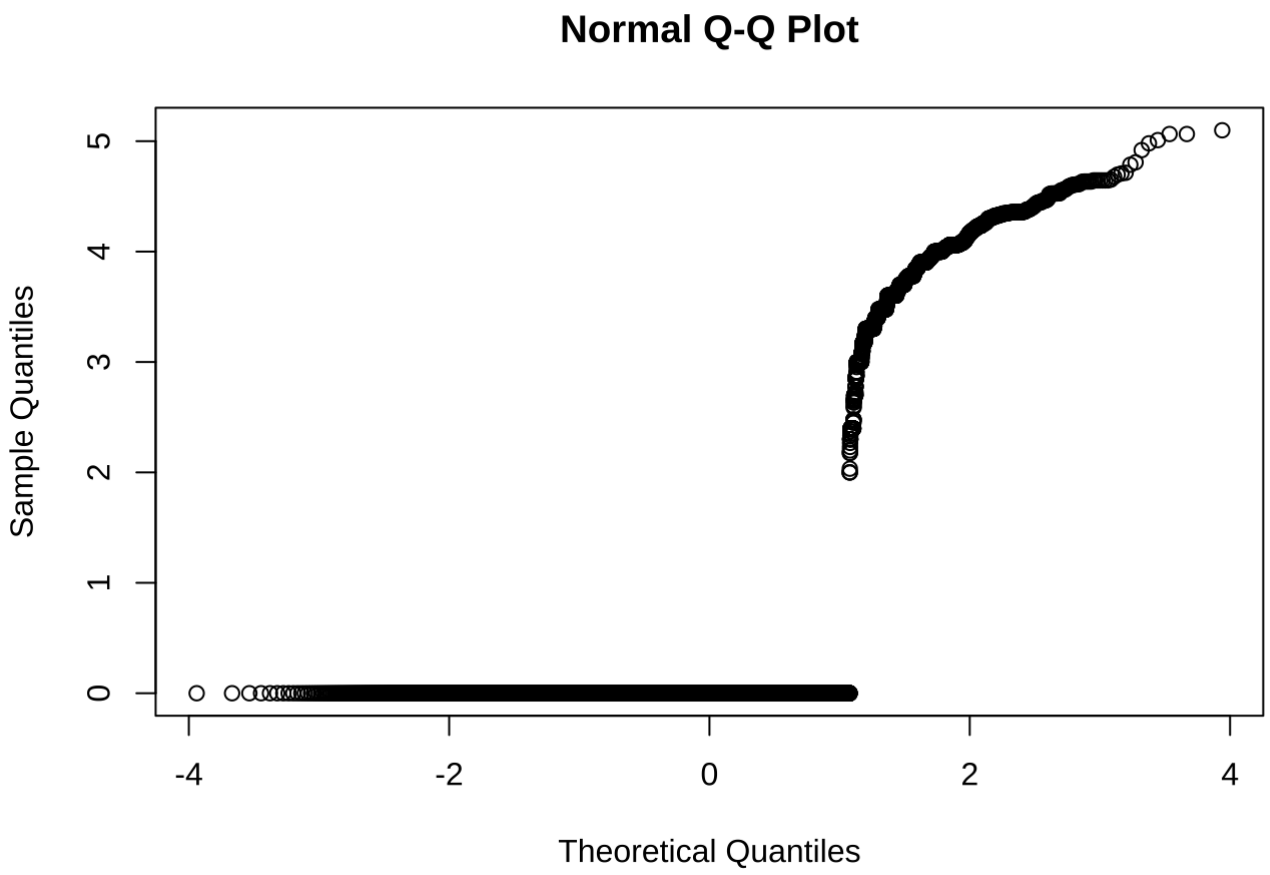
```
qqnorm(train_EDA$total_Work_Study)
```




```
ks.test(scale(train_EDA$total_Scholarship),"pnorm")
```

```
Warning in ks.test.default(scale(train_EDA$total_Scholarship), "pnorm"): ties
should not be present for the Kolmogorov-Smirnov test


    Asymptotic one-sample Kolmogorov-Smirnov test

data:  scale(train_EDA$total_Scholarship)
D = 0.51469, p-value < 2.2e-16
alternative hypothesis: two-sided
```

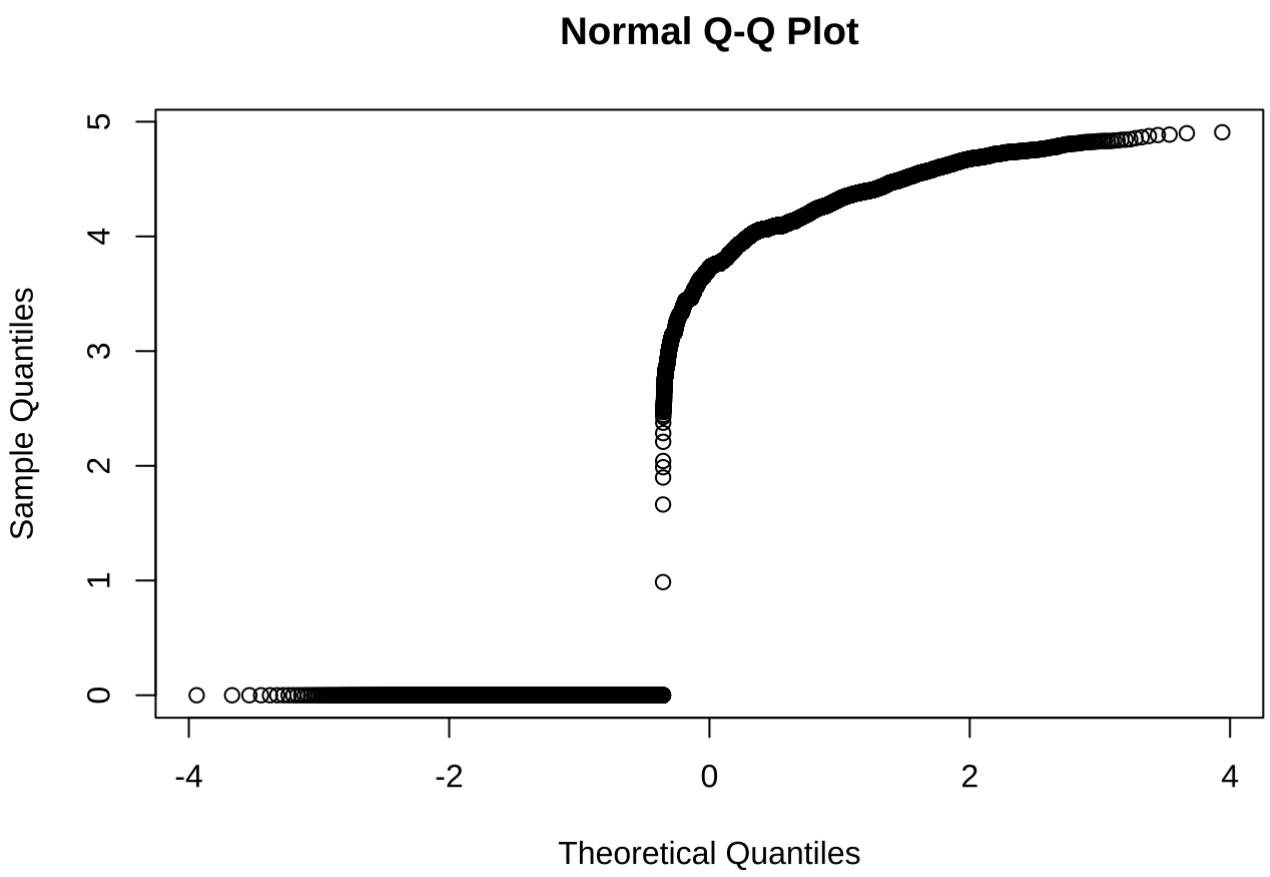
```
qqnorm(train_EDA$total_Scholarship)
```




```
ks.test(scale(train_EDA$total_Grant),"pnorm")
```

```
Warning in ks.test.default(scale(train_EDA$total_Grant), "pnorm"): ties should
not be present for the Kolmogorov-Smirnov test


    Asymptotic one-sample Kolmogorov-Smirnov test

data:  scale(train_EDA$total_Grant)
D = 0.26553, p-value < 2.2e-16
alternative hypothesis: two-sided
```

```
qqnorm(train_EDA$total_Grant)
```




```
####  Mann-Whitney U test
wilcox.test(train_EDA$total_Loan~train_EDA$Dropout)
```

```
    Wilcoxon rank sum test with continuity correction

data:  train_EDA$total_Loan by train_EDA$Dropout
W = 21246872, p-value < 2.2e-16
alternative hypothesis: true location shift is not equal to 0
```

```
wilcox.test(train_EDA$total_Scholarship~train_EDA$Dropout)
```

```
    Wilcoxon rank sum test with continuity correction

data:  train_EDA$total_Scholarship by train_EDA$Dropout
W = 20238064, p-value < 2.2e-16
alternative hypothesis: true location shift is not equal to 0
```

```
wilcox.test(train_EDA$total_Work_Study~train_EDA$Dropout)
```

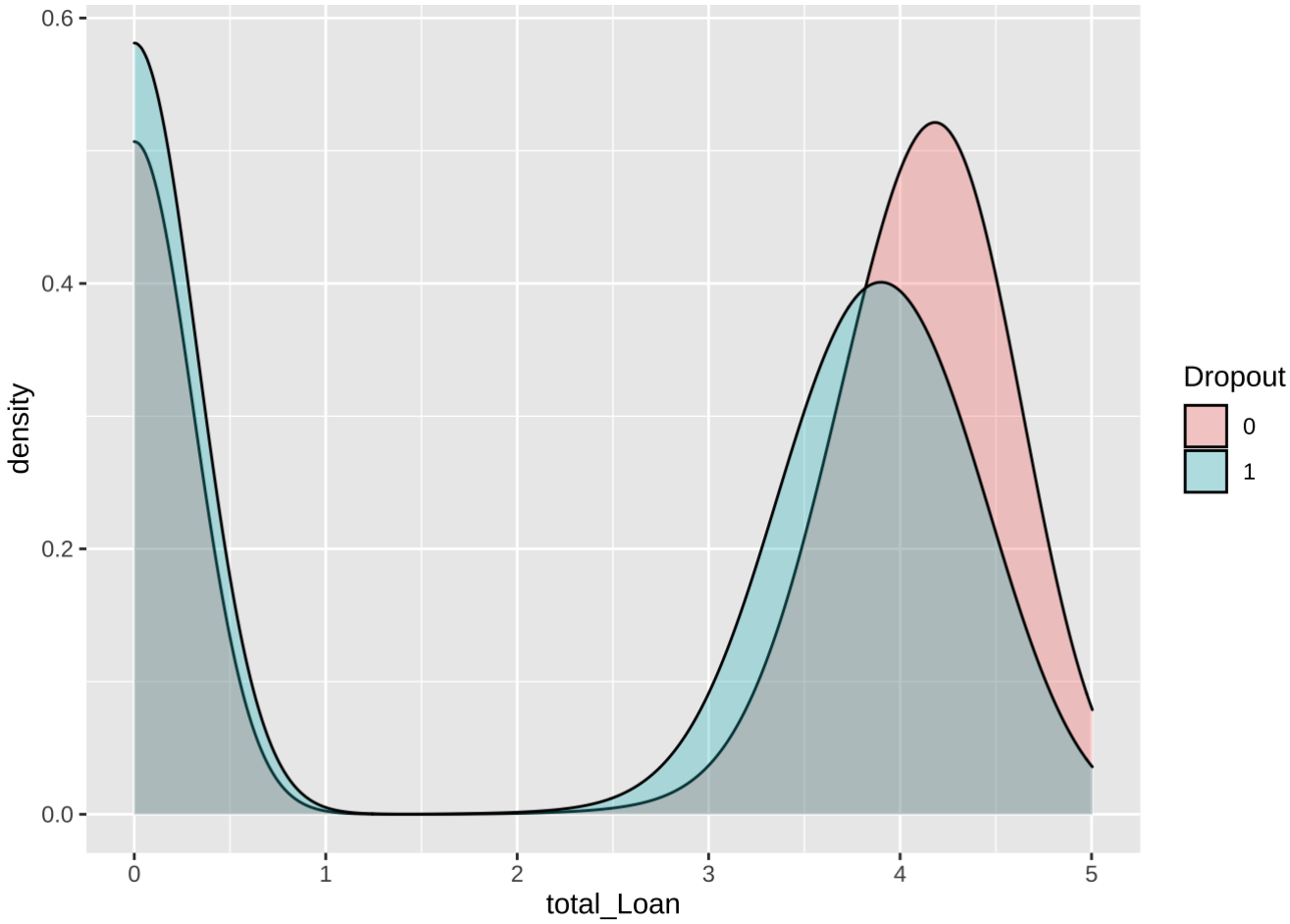
```
    Wilcoxon rank sum test with continuity correction

data:  train_EDA$total_Work_Study by train_EDA$Dropout
W = 18465810, p-value = 2.475e-15
alternative hypothesis: true location shift is not equal to 0
```

```
wilcox.test(train_EDA$total_Grant~train_EDA$Dropout)
```

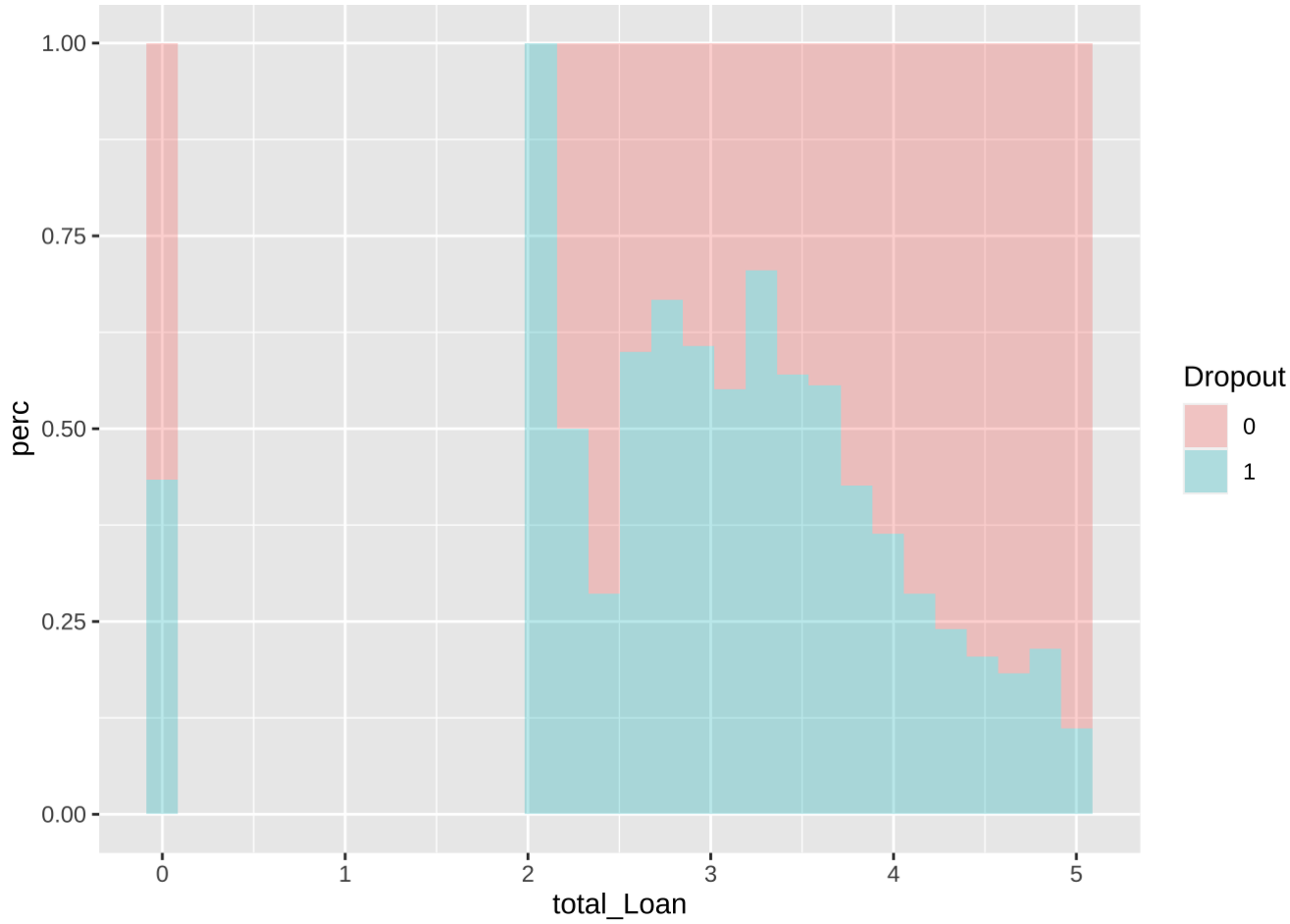
```
    Wilcoxon rank sum test with continuity correction

data:  train_EDA$total_Grant by train_EDA$Dropout
W = 20877692, p-value < 2.2e-16
alternative hypothesis: true location shift is not equal to 0
```

```
##### plot
for(i in c("total_Loan","total_Scholarship","total_Work_Study","total_Grant")){
print(ggplot(train_EDA, aes(x = (get(i)), fill = Dropout)) +xlab(i)+geom_density(alpha =
print(ggplot(data = train_EDA, mapping = aes(x = get(i), fill =Dropout)) +xlab(i)+ ylab("
}
```

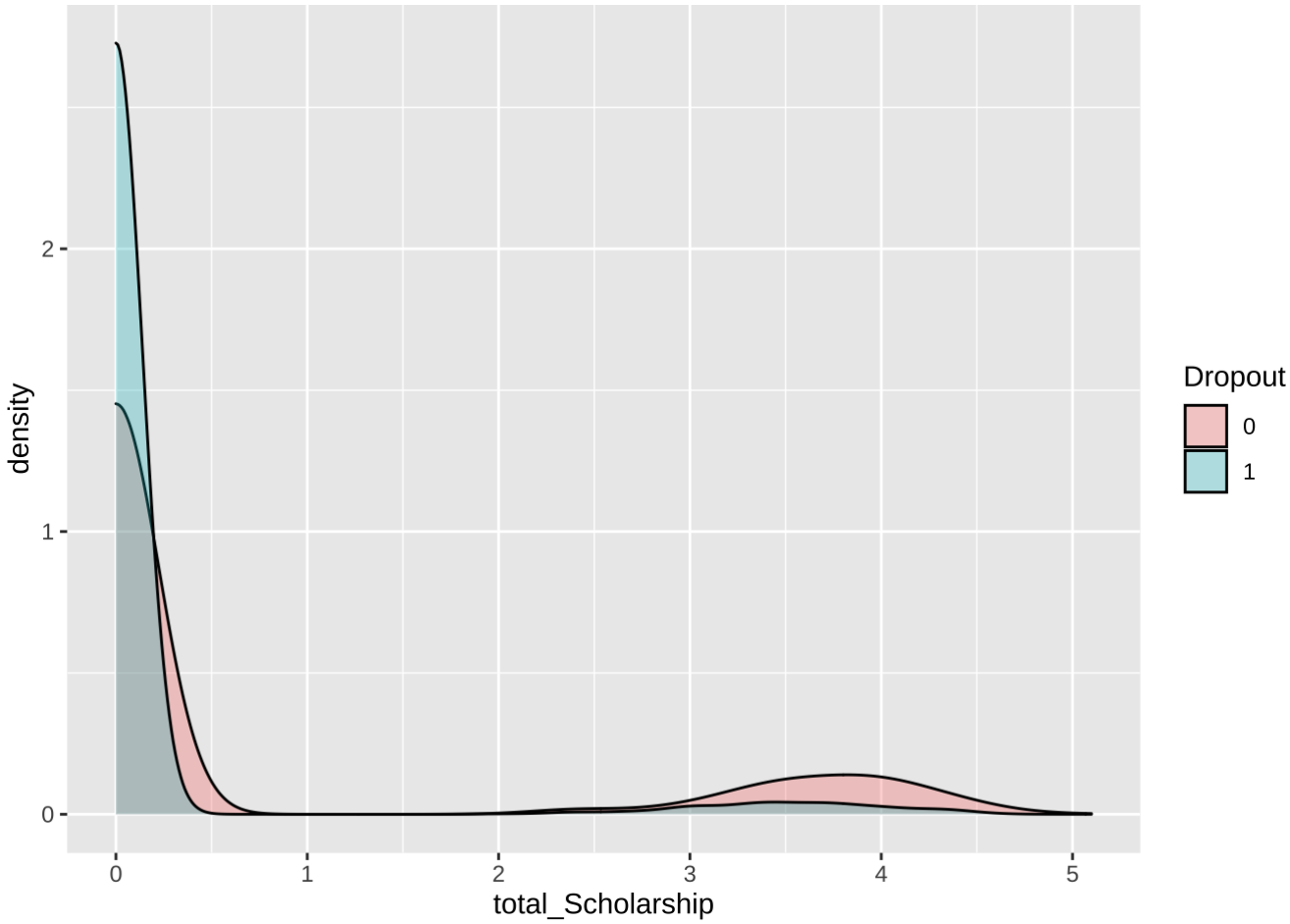
```
Warning: Removed 1 rows containing non-finite values (stat_density).
```



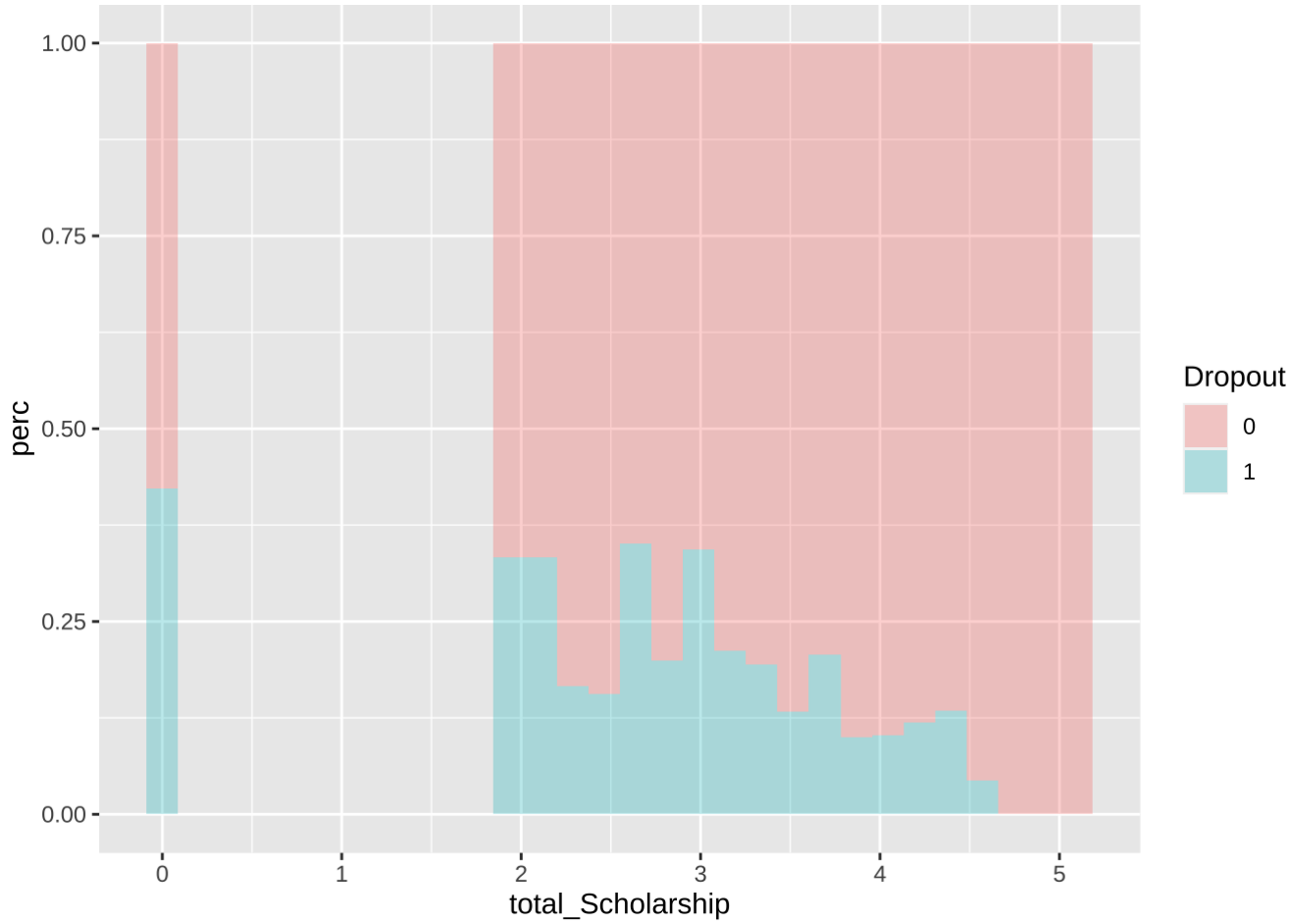

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

Warning: Removed 1 rows containing non-finite values (stat_bin).

Warning: Removed 22 rows containing missing values (geom_bar).
```



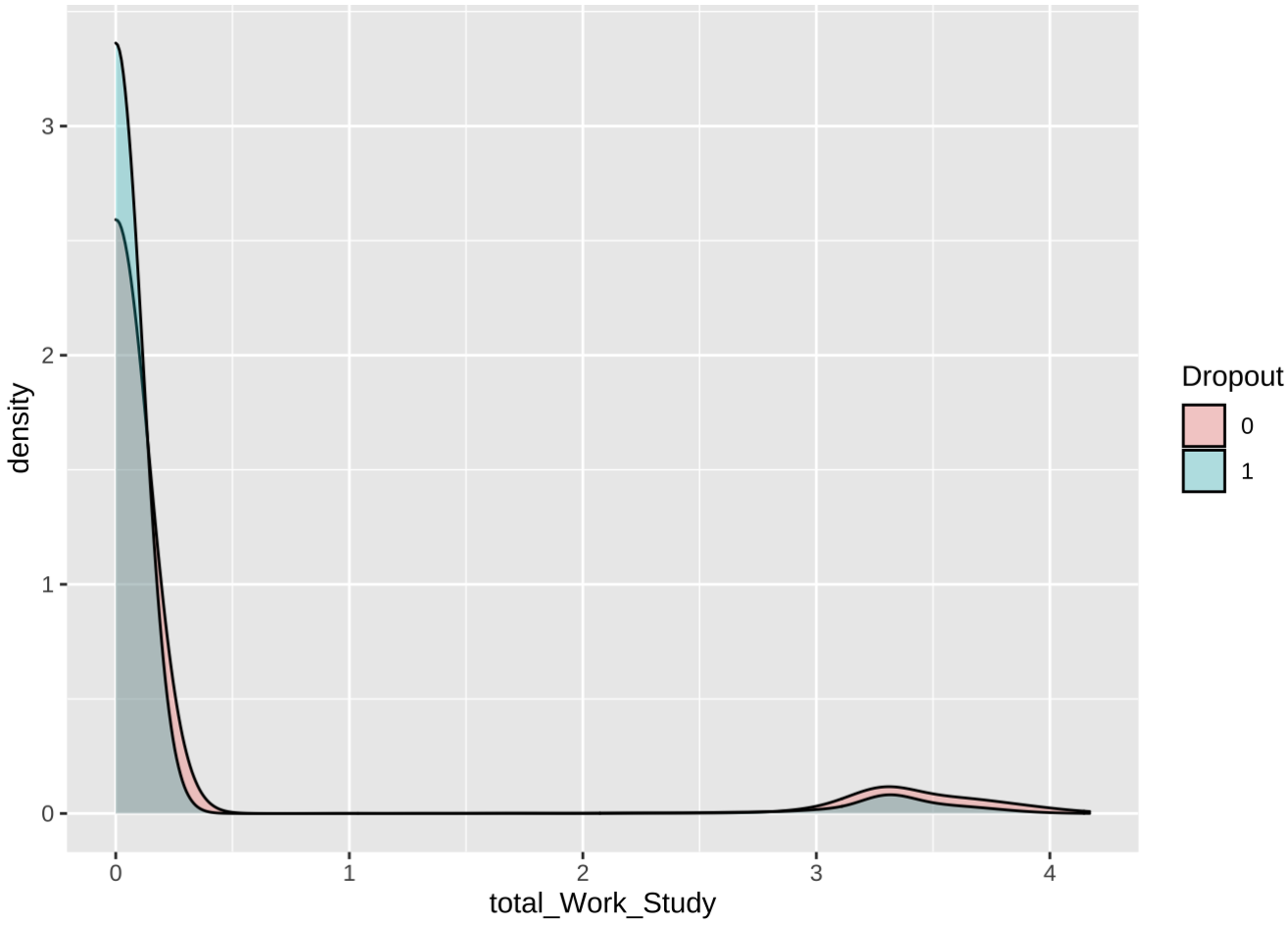

```
Warning: Removed 1 rows containing non-finite values (stat_density).
```




```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

Warning: Removed 1 rows containing non-finite values (stat_bin).

Warning: Removed 20 rows containing missing values (geom_bar).
```



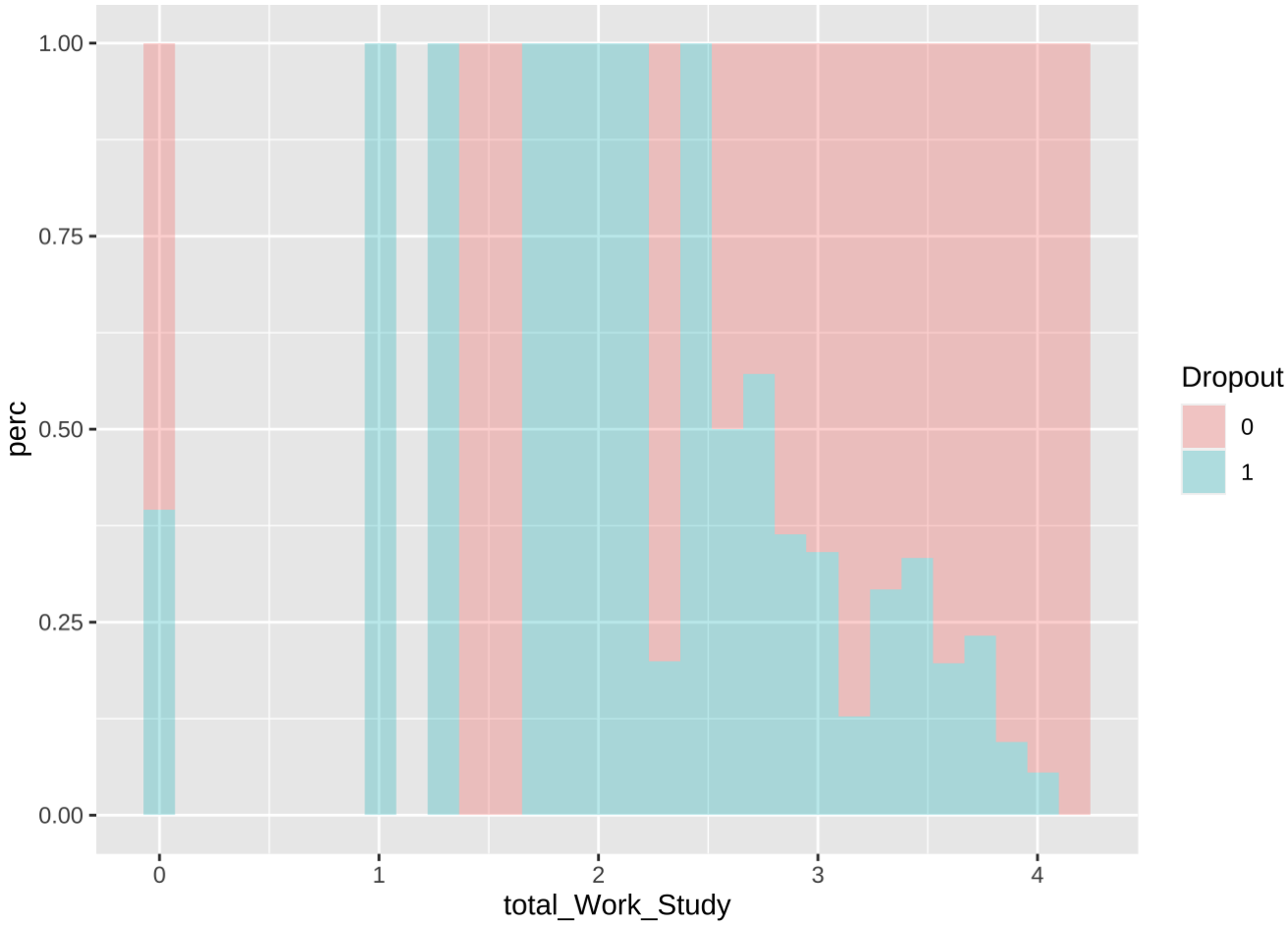

```
Warning: Removed 1 rows containing non-finite values (stat_density).
```



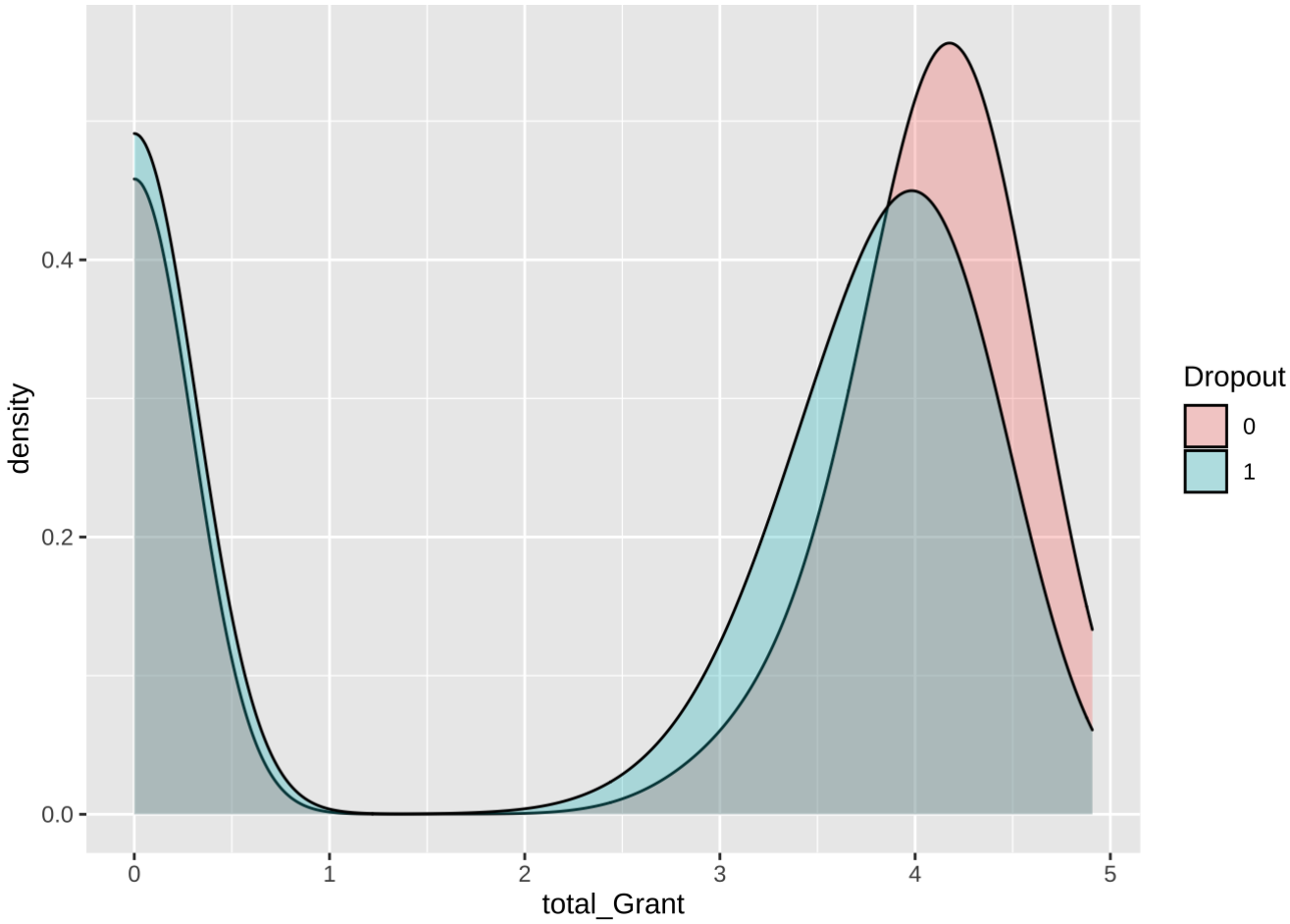

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
Warning: Removed 1 rows containing non-finite values (stat_bin).
```

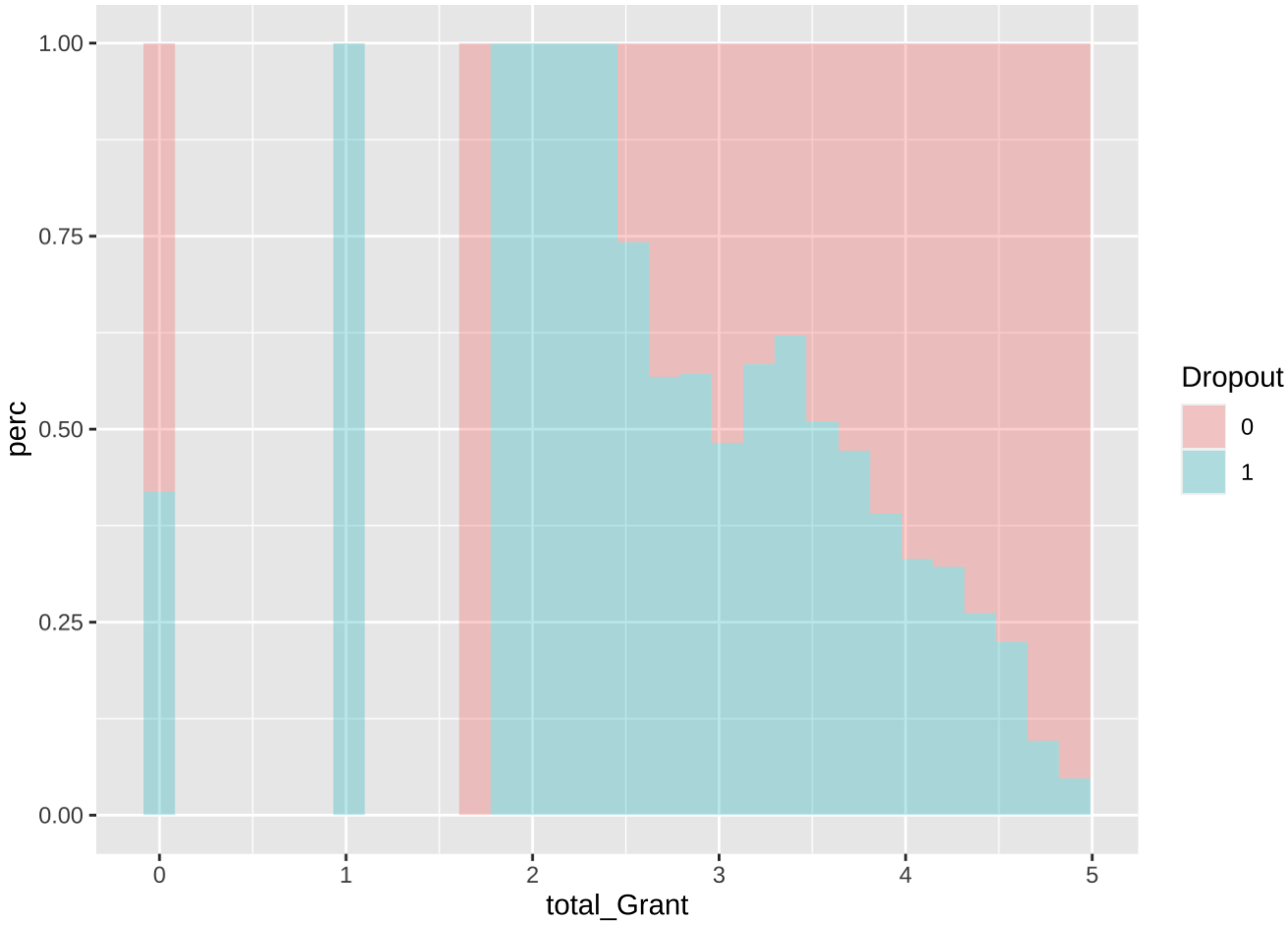
```
Warning: Removed 14 rows containing missing values (geom_bar).
```




```
Warning: Removed 1 rows containing non-finite values (stat_density).
```




```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

Warning: Removed 1 rows containing non-finite values (stat_bin).

Warning: Removed 16 rows containing missing values (geom_bar).
```



Analysis and Main Results:

None of these four variables are normally distributed, so Wilcoxon rank sum test was conducted and results show that all these three variables are significantly correlated with dropout.

So all these four variables are left and could be further selected (Loan|Work.Study|Grant|Scholarship) in the model

Involved variables:

- Loan
- Work.Study
- Grant
- Scholarship

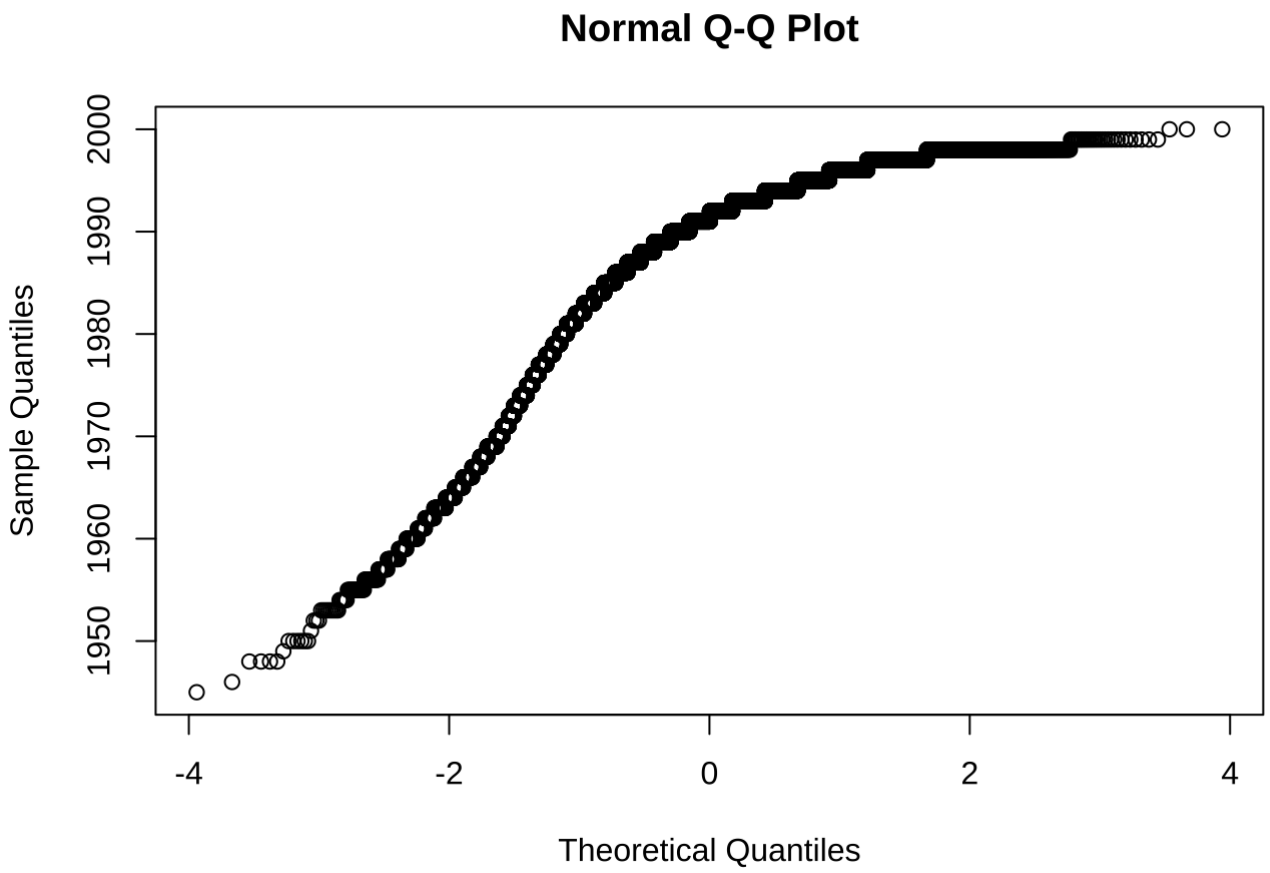
BirthYear, HSDipYr, enrolled_age

```
############  normal distribution test
ks.test(scale(train_EDA$BirthYear),"pnorm")
```

```
Warning in ks.test.default(scale(train_EDA$BirthYear), "pnorm"): ties should not
be present for the Kolmogorov-Smirnov test
```

```
    Asymptotic one-sample Kolmogorov-Smirnov test

data:  scale(train_EDA$BirthYear)
D = 0.17196, p-value < 2.2e-16
alternative hypothesis: two-sided
```

```
qqnorm(train_EDA$BirthYear)
```



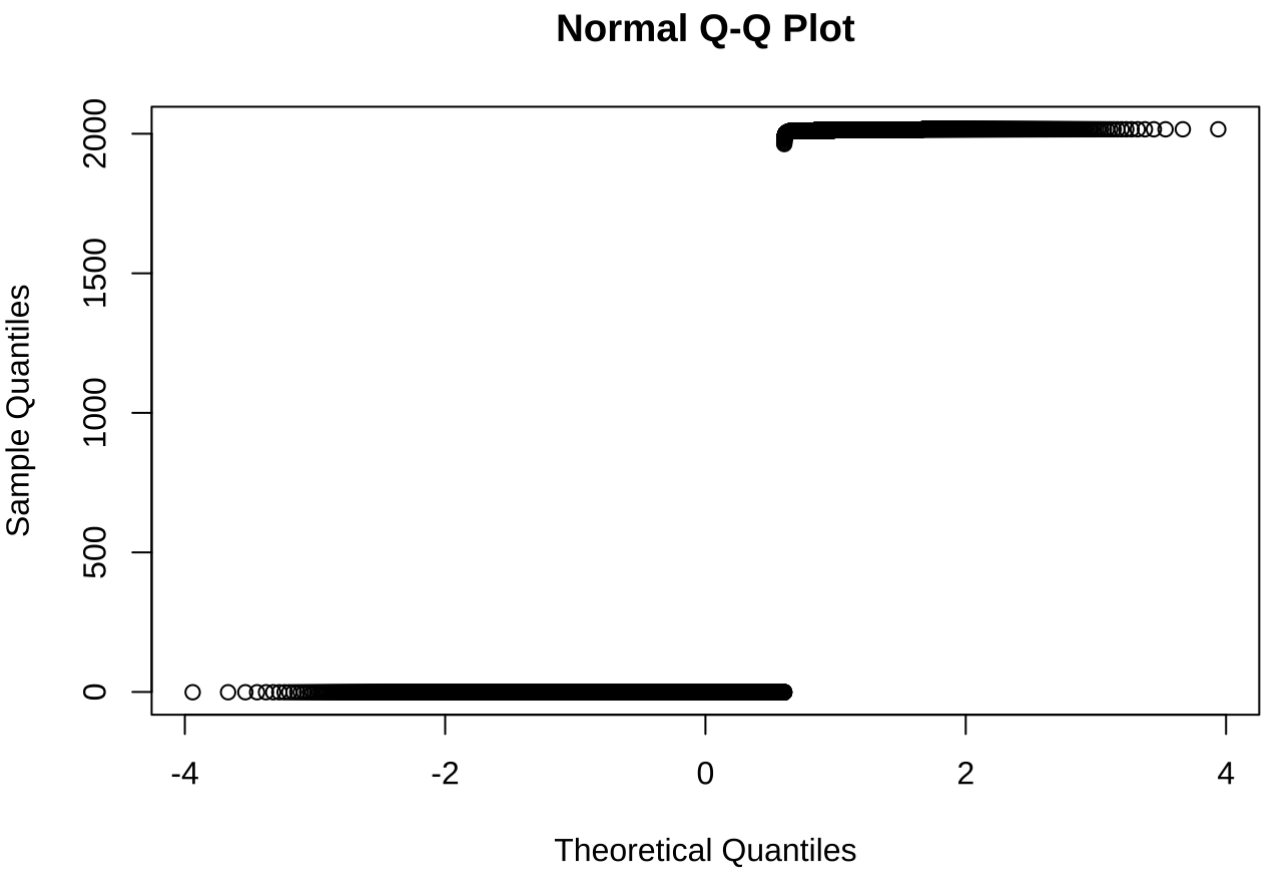

```
ks.test(scale(train_EDA$HSDipYr),"pnorm")
```

```
Warning in ks.test.default(scale(train_EDA$HSDipYr), "pnorm"): ties should not
be present for the Kolmogorov-Smirnov test
```

```
    Asymptotic one-sample Kolmogorov-Smirnov test

data:  scale(train_EDA$HSDipYr)
D = 0.45726, p-value < 2.2e-16
alternative hypothesis: two-sided
```

```
alternative hypothesis: two-sided
```

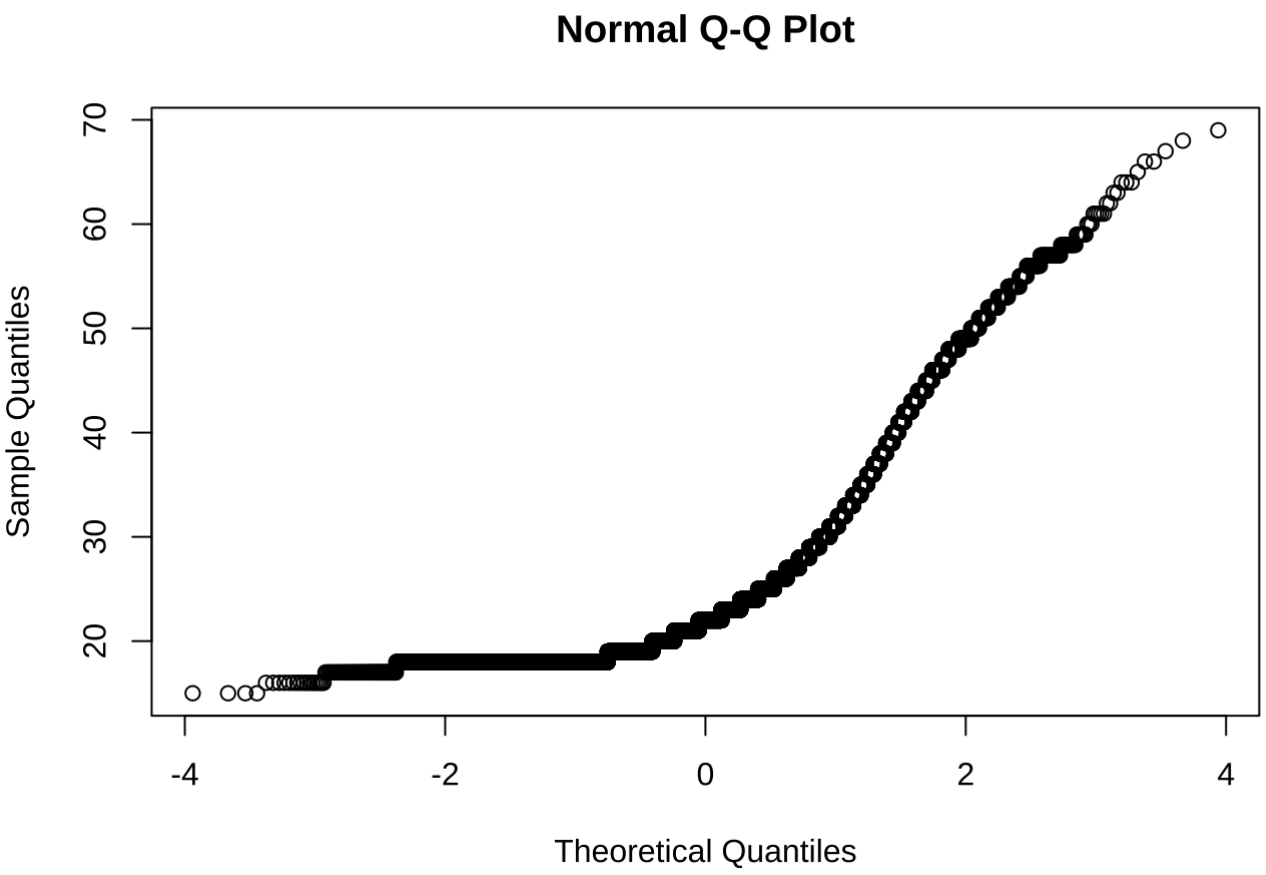
```
qqnorm(train_EDA$HSDipYr)
```




```
ks.test(scale(train_EDA$enrolled_age),"pnorm")
```

```
Warning in ks.test.default(scale(train_EDA$enrolled_age), "pnorm"): ties should
not be present for the Kolmogorov-Smirnov test


    Asymptotic one-sample Kolmogorov-Smirnov test

data:  scale(train_EDA$enrolled_age)
D = 0.20176, p-value < 2.2e-16
alternative hypothesis: two-sided
```

```
qqnorm(train_EDA$enrolled_age)
```




```
####  Mann-Whitney U test
wilcox.test(train_EDA$BirthYear~train_EDA$Dropout)
```

```
    Wilcoxon rank sum test with continuity correction

data:  train_EDA$BirthYear by train_EDA$Dropout
W = 20817719, p-value < 2.2e-16
alternative hypothesis: true location shift is not equal to 0
```

```
wilcox.test(train_EDA$HSDipYr~train_EDA$Dropout)
```

```
    Wilcoxon rank sum test with continuity correction

data:  train_EDA$HSDipYr by train_EDA$Dropout
W = 18119246, p-value = 0.04035
alternative hypothesis: true location shift is not equal to 0
```

```
wilcox.test(train_EDA$enrolled_age~train_EDA$Dropout)
```

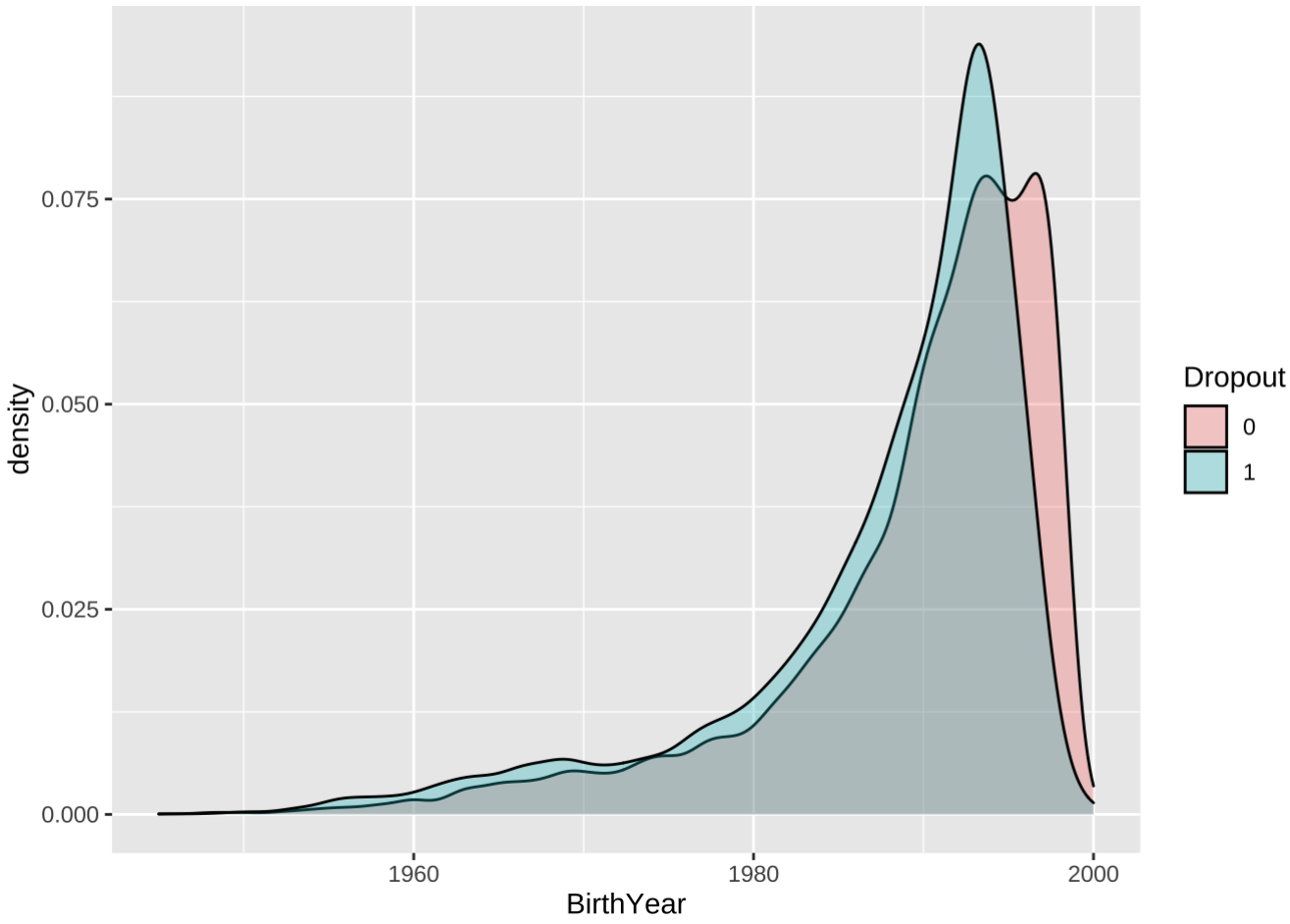
```
    Wilcoxon rank sum test with continuity correction

data:  train_EDA$enrolled_age by train_EDA$Dropout
W = 17267425, p-value = 0.004015
alternative hypothesis: true location shift is not equal to 0
```

```
sum(is.na(train_EDA$enrolled_age))
```

```
[1] 1
```

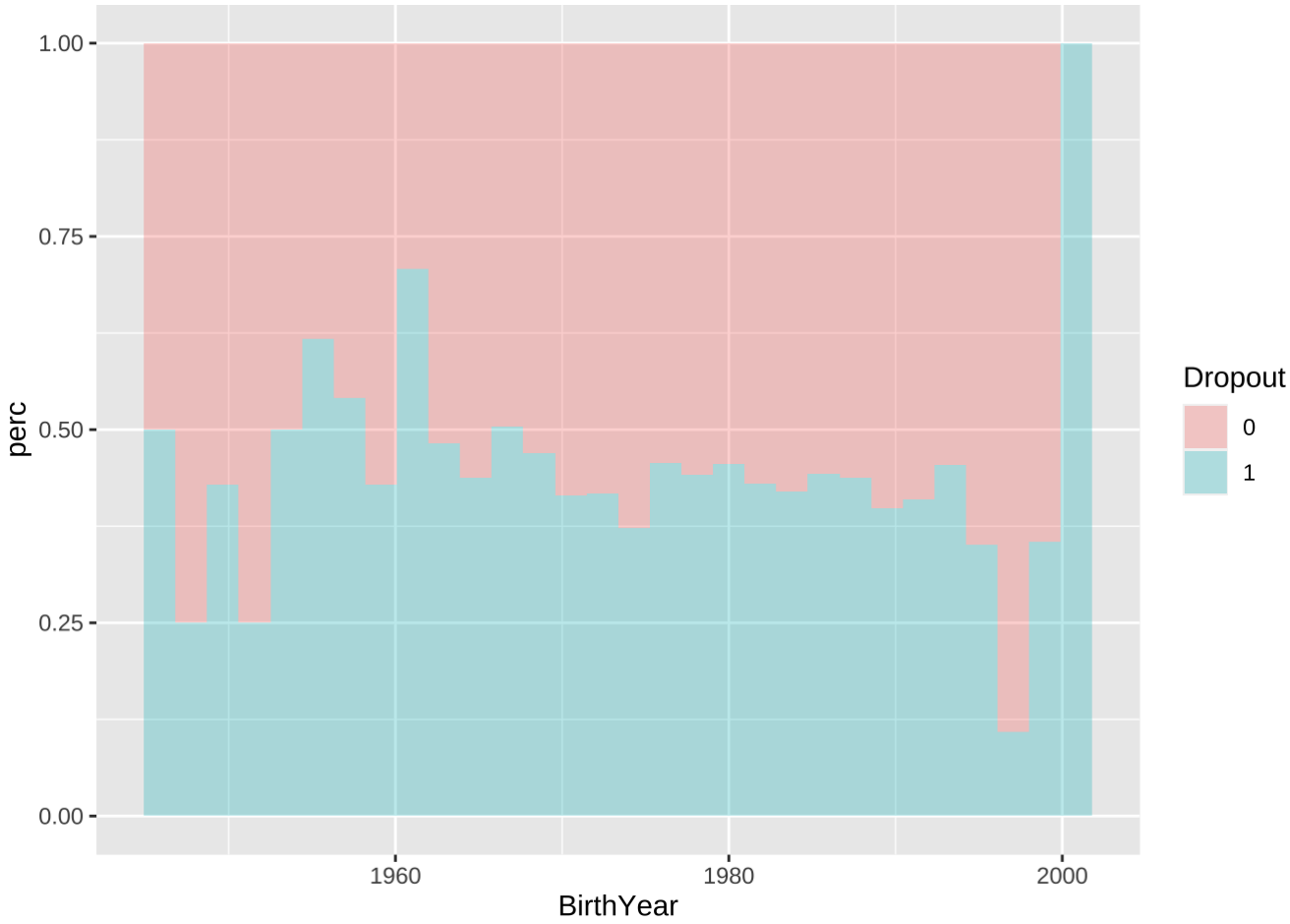
```
for(i in c("BirthYear","HSDipYr","enrolled_age")){
print(ggplot(train_EDA, aes(x = (get(i)), fill = Dropout)) +xlab(i)+geom_density(alpha =
print(ggplot(data = train_EDA, mapping = aes(x = get(i), fill =Dropout)) +xlab(i)+ ylab("
}
```

```
Warning: Removed 1 rows containing non-finite values (stat_density).
```

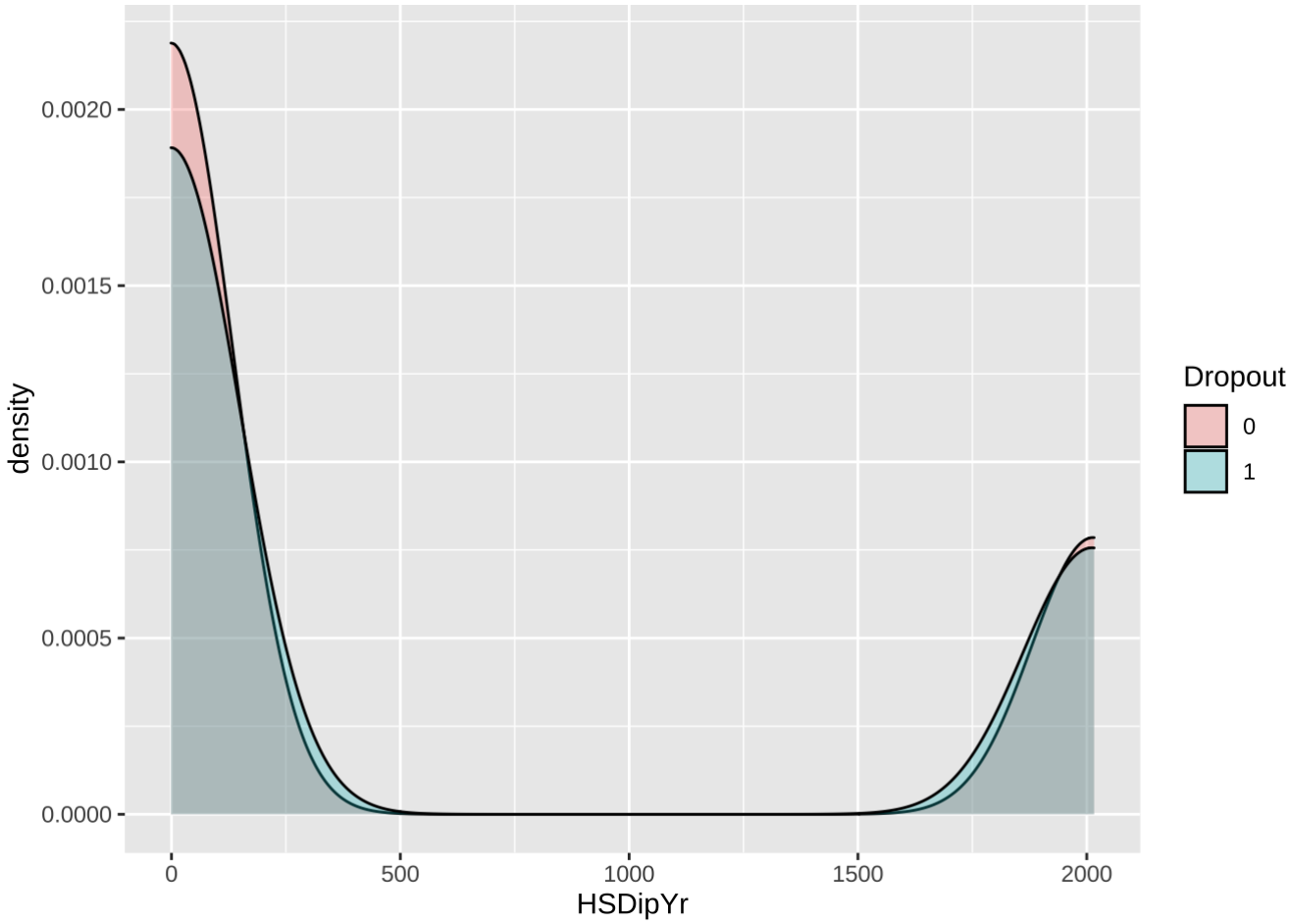



```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

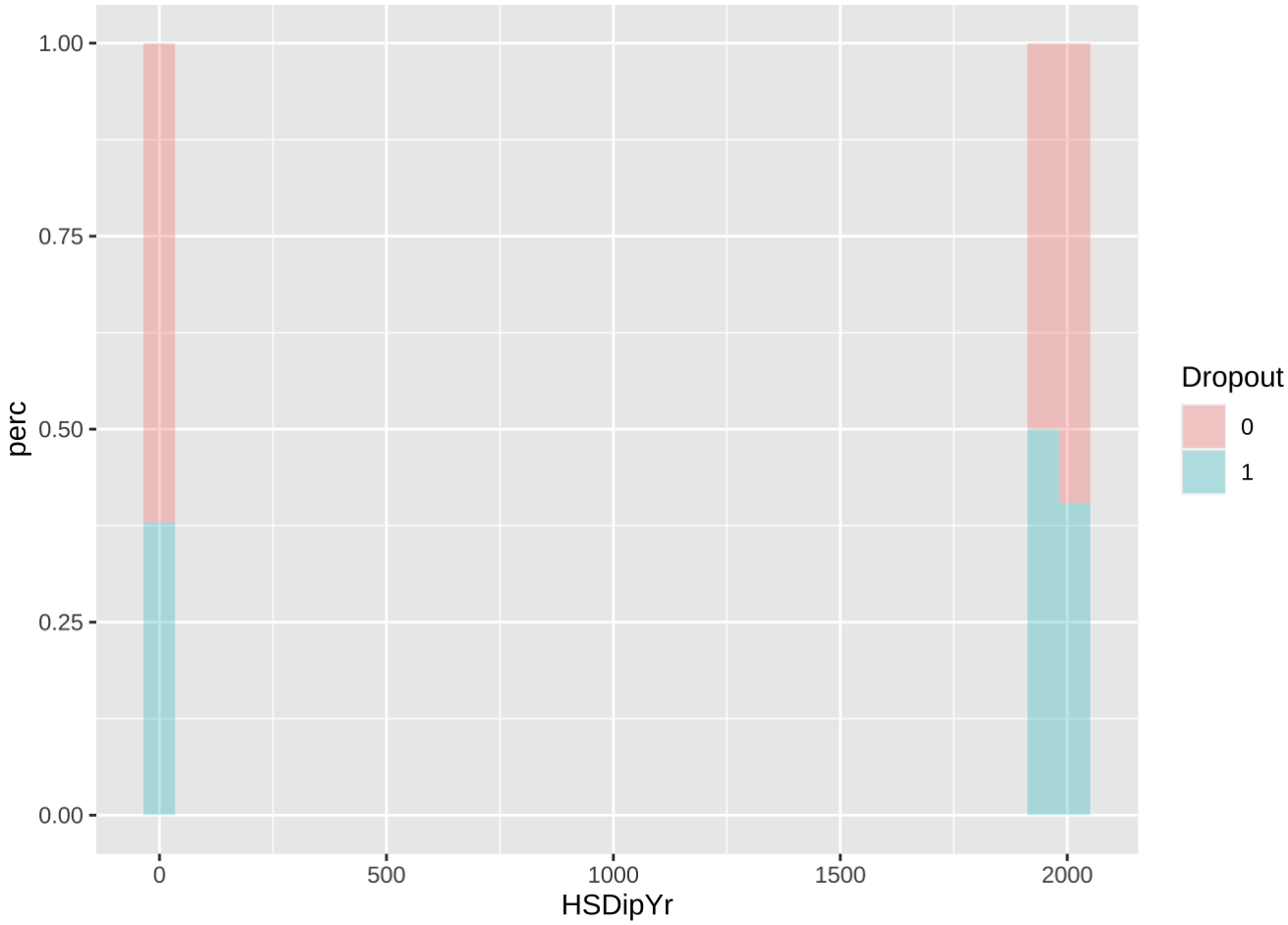
```
Warning: Removed 1 rows containing non-finite values (stat_bin).
```



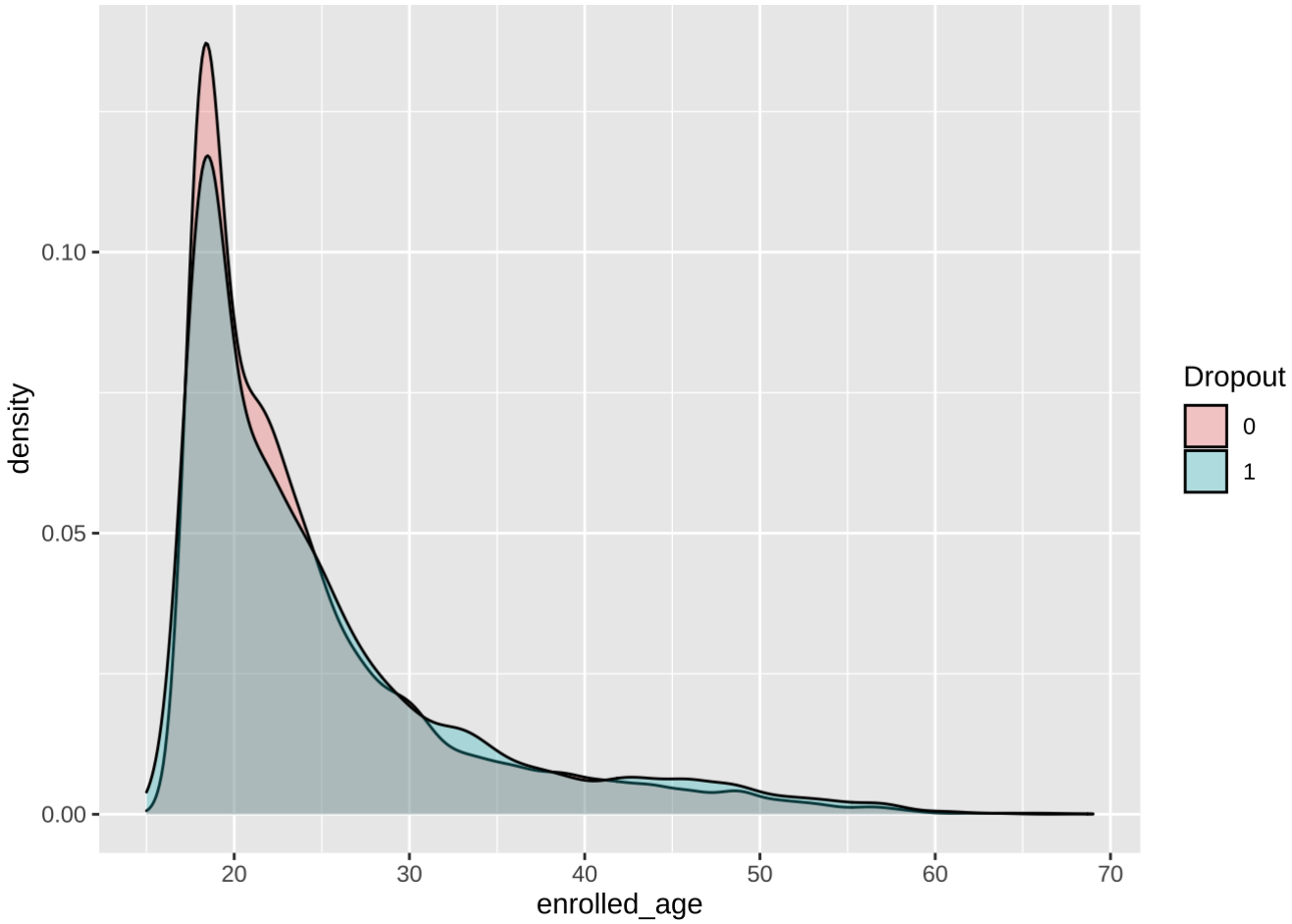

```
Warning: Removed 1 rows containing non-finite values (stat_density).
```




```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

Warning: Removed 1 rows containing non-finite values (stat_bin).

Warning: Removed 54 rows containing missing values (geom_bar).
```

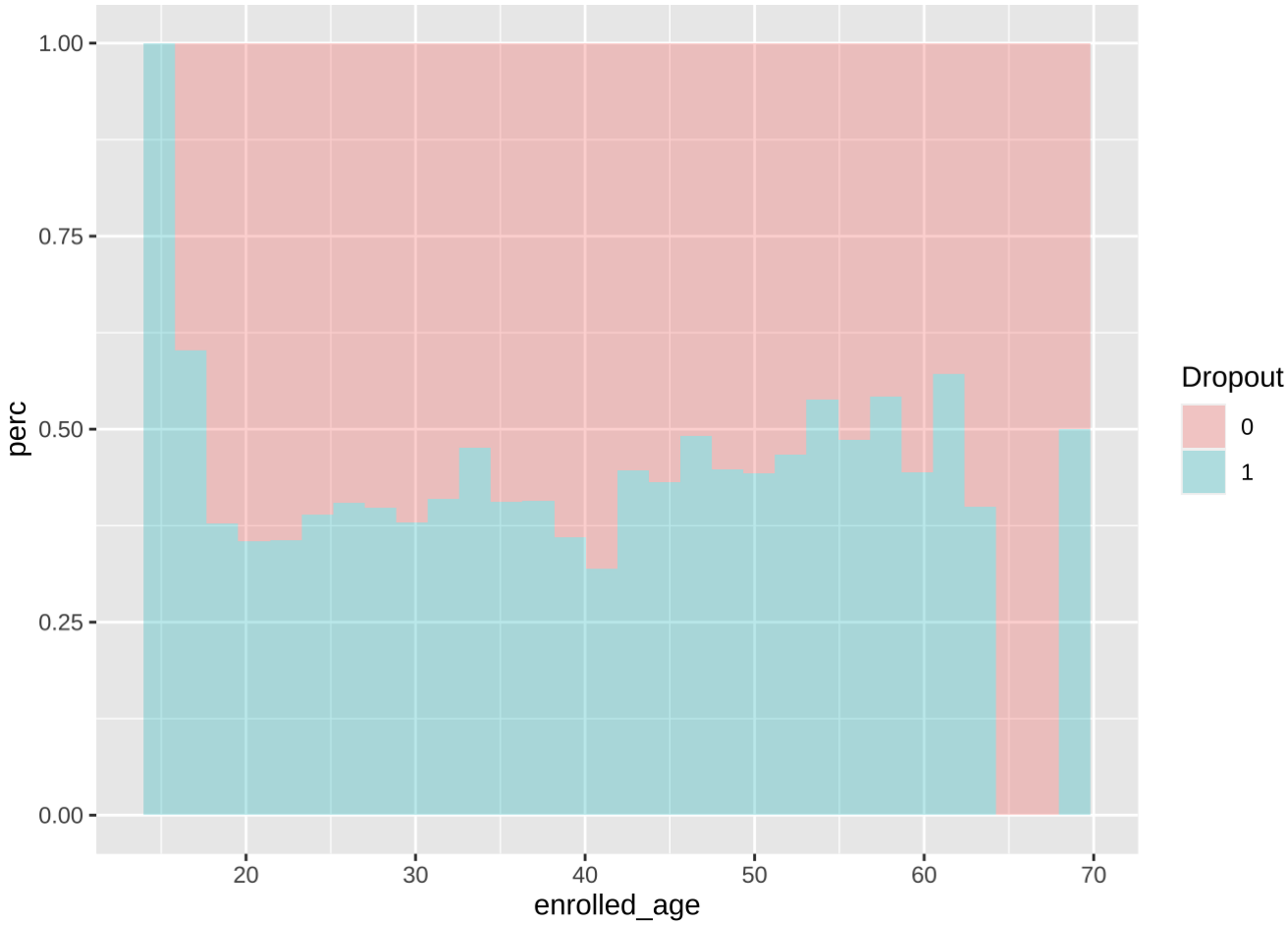



```
Warning: Removed 1 rows containing non-finite values (stat_density).
```




```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
Warning: Removed 1 rows containing non-finite values (stat_bin).
```



Analysis and Main Results:

None of them are normally distributed, so Wilcoxon rank sum test was conducted and results show that BirthYear is significantly correlated with dropout while the correlation between HSDipYr,enrolled_age and dropout is not significant at 0.001 level

So BirthYear will be involved and could be further selected in the model.

Involved variables:

- BirthYear

```
# df<-df[,-"enrolled_age","HSDipYr")]
df <- df %>% select(-c(enrolled_age, HSDipYr))
```

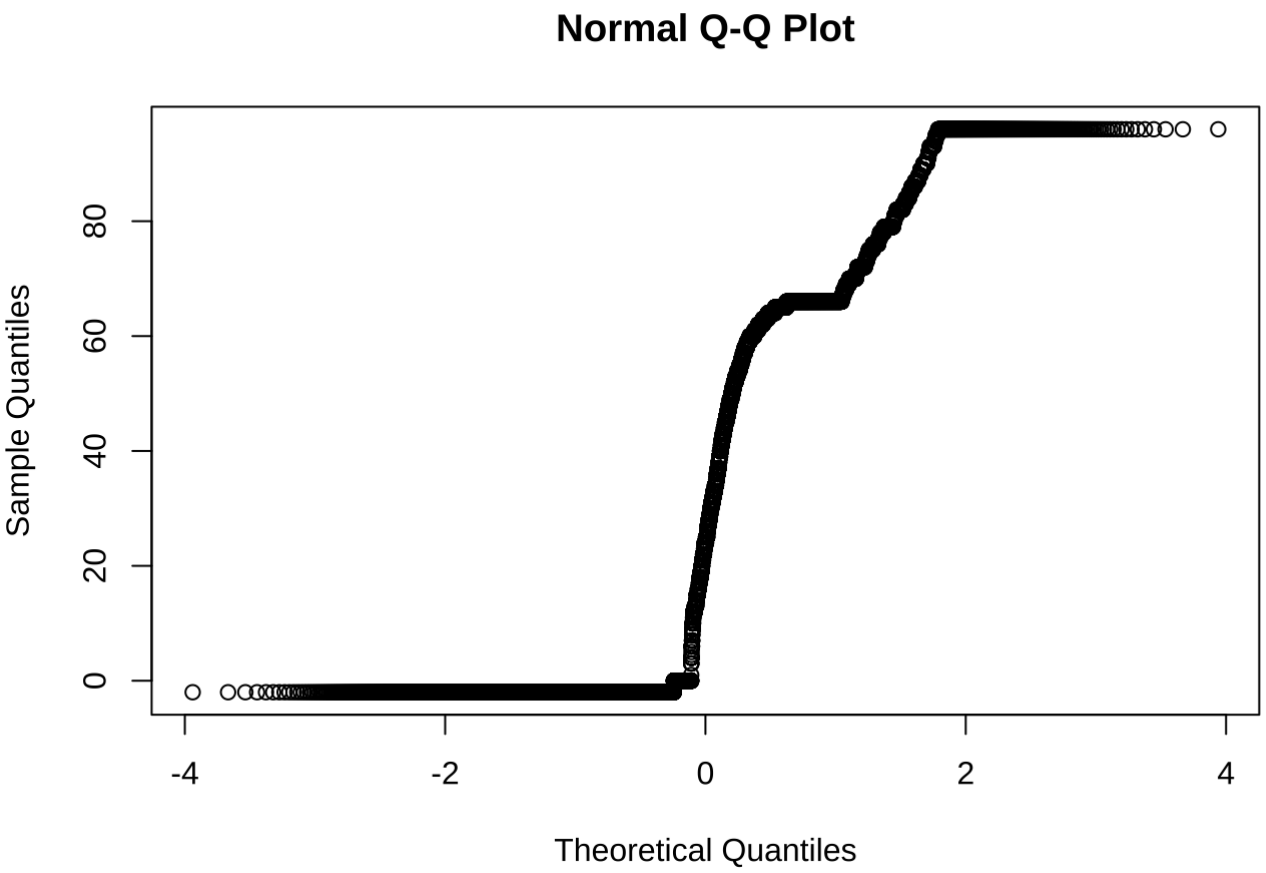
NumColCredAcceptTransfer & NumColCredAttemptTransfer

```
############  normal distribution test
ks.test(scale(train_EDA$NumColCredAcceptTransfer),"pnorm")
```

```
Warning in ks.test.default(scale(train_EDA$NumColCredAcceptTransfer), "pnorm"):
ties should not be present for the Kolmogorov-Smirnov test
```

```
    Asymptotic one-sample Kolmogorov-Smirnov test

data:  scale(train_EDA$NumColCredAcceptTransfer)
D = 0.28036, p-value < 2.2e-16
alternative hypothesis: two-sided
```

```
qqnorm(train_EDA$NumColCredAcceptTransfer)
```

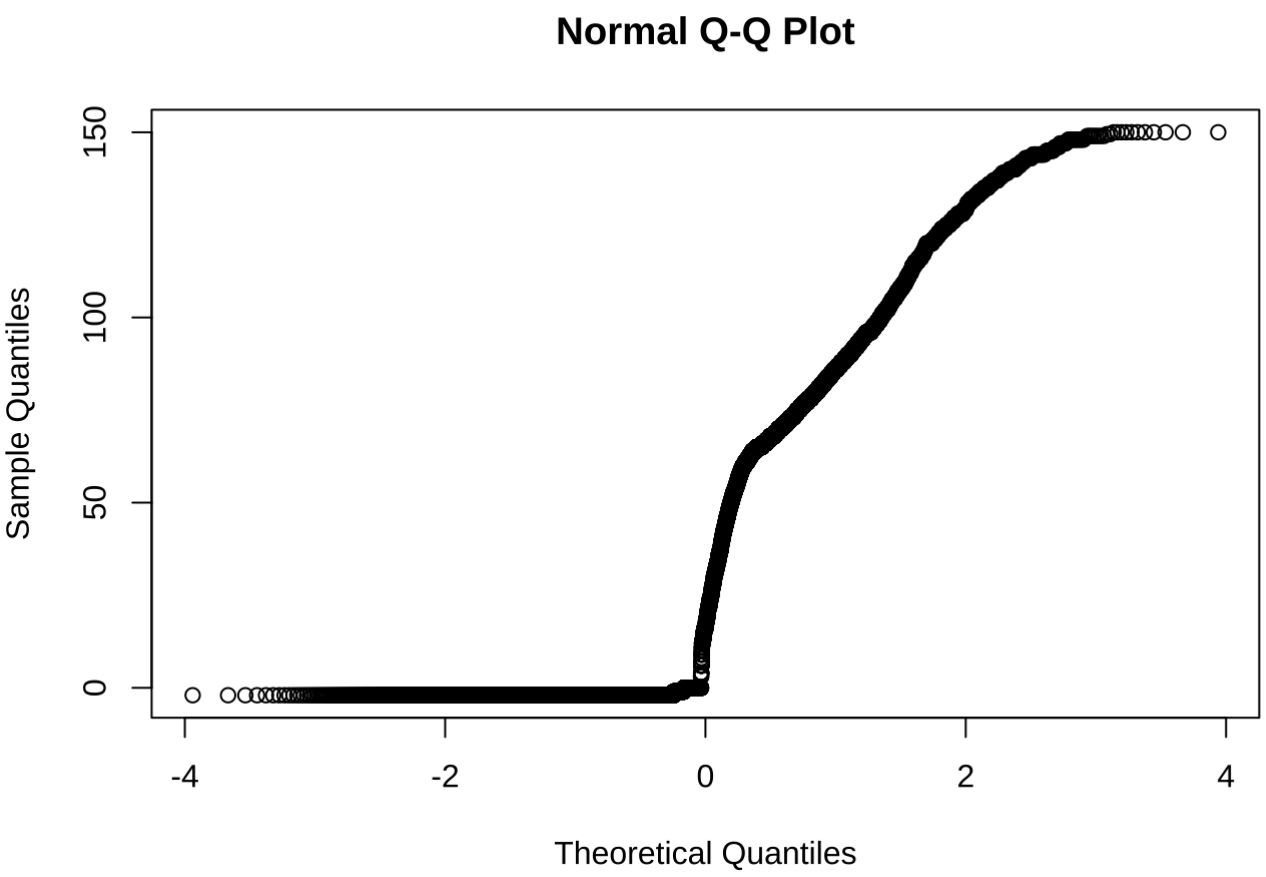



```
ks.test(scale(train_EDA$NumColCredAttemptTransfer),"pnorm")
```

```
Warning in ks.test.default(scale(train_EDA$NumColCredAttemptTransfer), "pnorm"):
ties should not be present for the Kolmogorov-Smirnov test
```

```
    Asymptotic one-sample Kolmogorov-Smirnov test

data:  scale(train_EDA$NumColCredAttemptTransfer)
D = 0.29205, p-value < 2.2e-16
alternative hypothesis: two-sided
```

```
qqnorm(train_EDA$NumColCredAttemptTransfer)
```




```
####  Mann-Whitney U test
wilcox.test(train_EDA$NumColCredAttemptTransfer~train_EDA$Dropout)
```

```
    Wilcoxon rank sum test with continuity correction

data:  train_EDA$NumColCredAttemptTransfer by train_EDA$Dropout
W = 19218175, p-value = 2.475e-14
alternative hypothesis: true location shift is not equal to 0
```

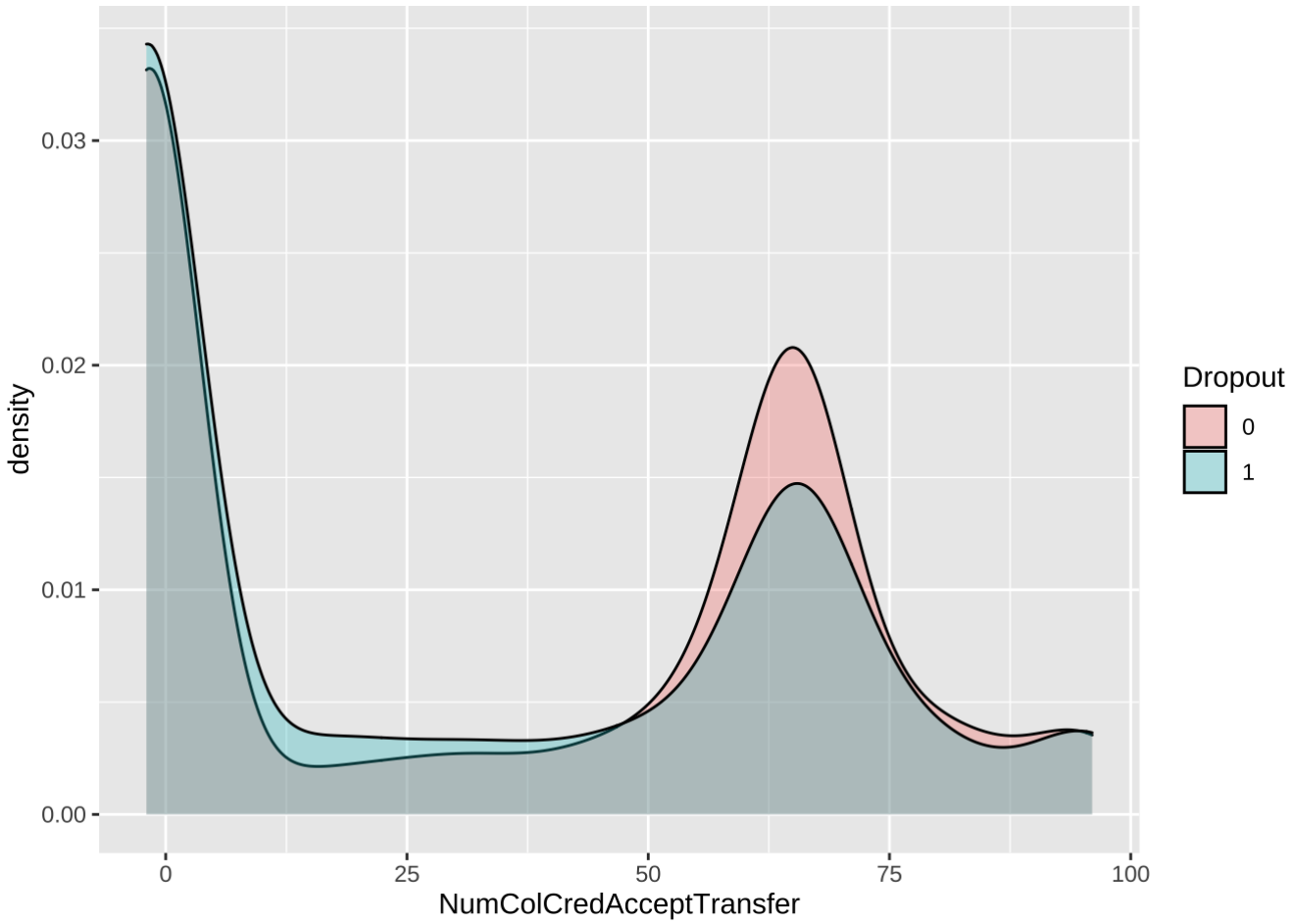
```
wilcox.test(train_EDA$NumColCredAttemptTransfer~train_EDA$Dropout)
```

```
    Wilcoxon rank sum test with continuity correction

data:  train_EDA$NumColCredAttemptTransfer by train_EDA$Dropout
W = 19218175, p-value = 2.475e-14
alternative hypothesis: true location shift is not equal to 0
```

```
for(i in c("NumColCredAcceptTransfer","NumColCredAttemptTransfer")){
print(ggplot(train_EDA, aes(x = (get(i)), fill = Dropout)) +xlab(i)+geom_density(alpha =
print(ggplot(data = train_EDA, mapping = aes(x = get(i), fill =Dropout)) +xlab(i)+ ylab("
```

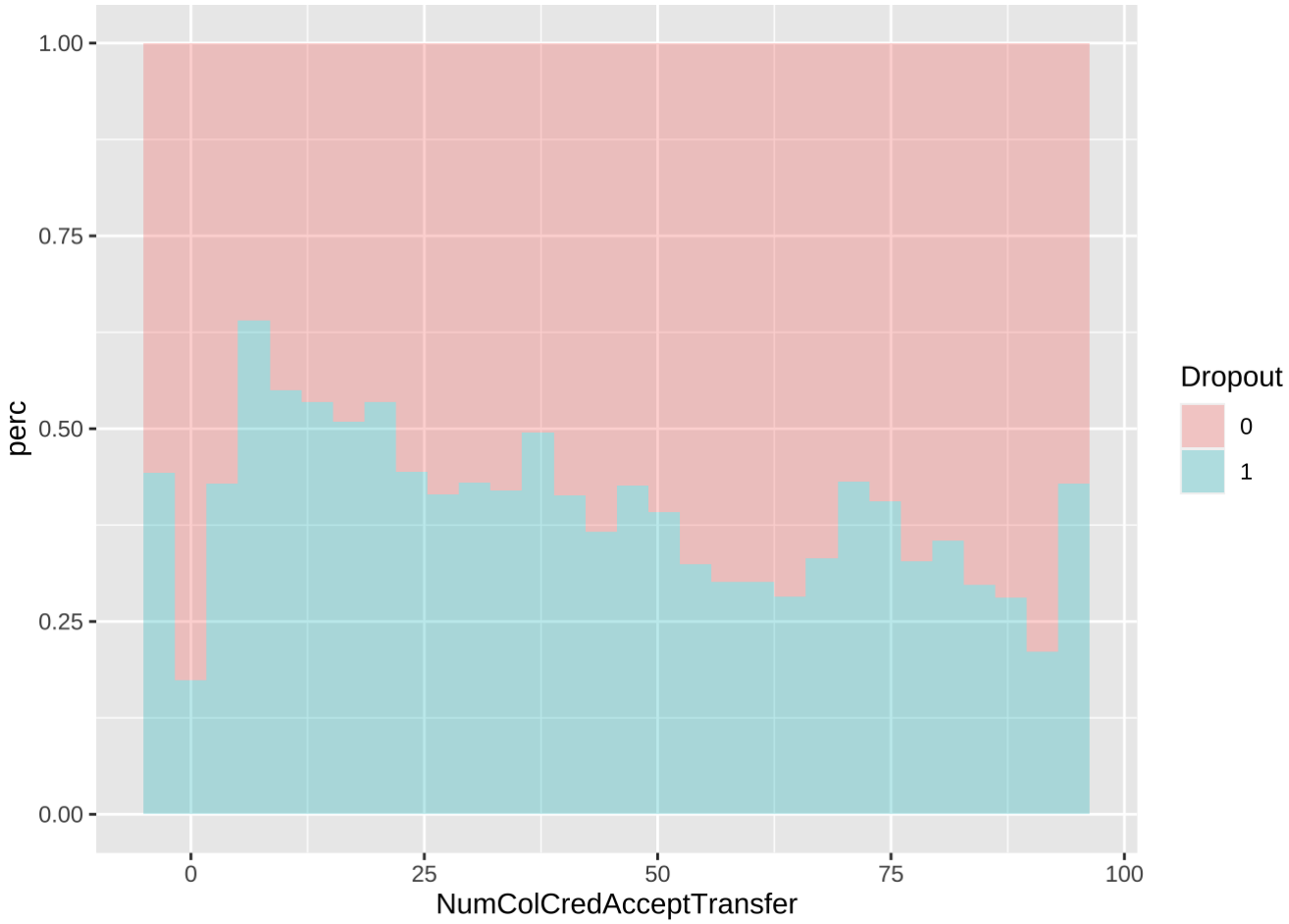
```
}
```

```
Warning: Removed 1 rows containing non-finite values (stat_density).
```

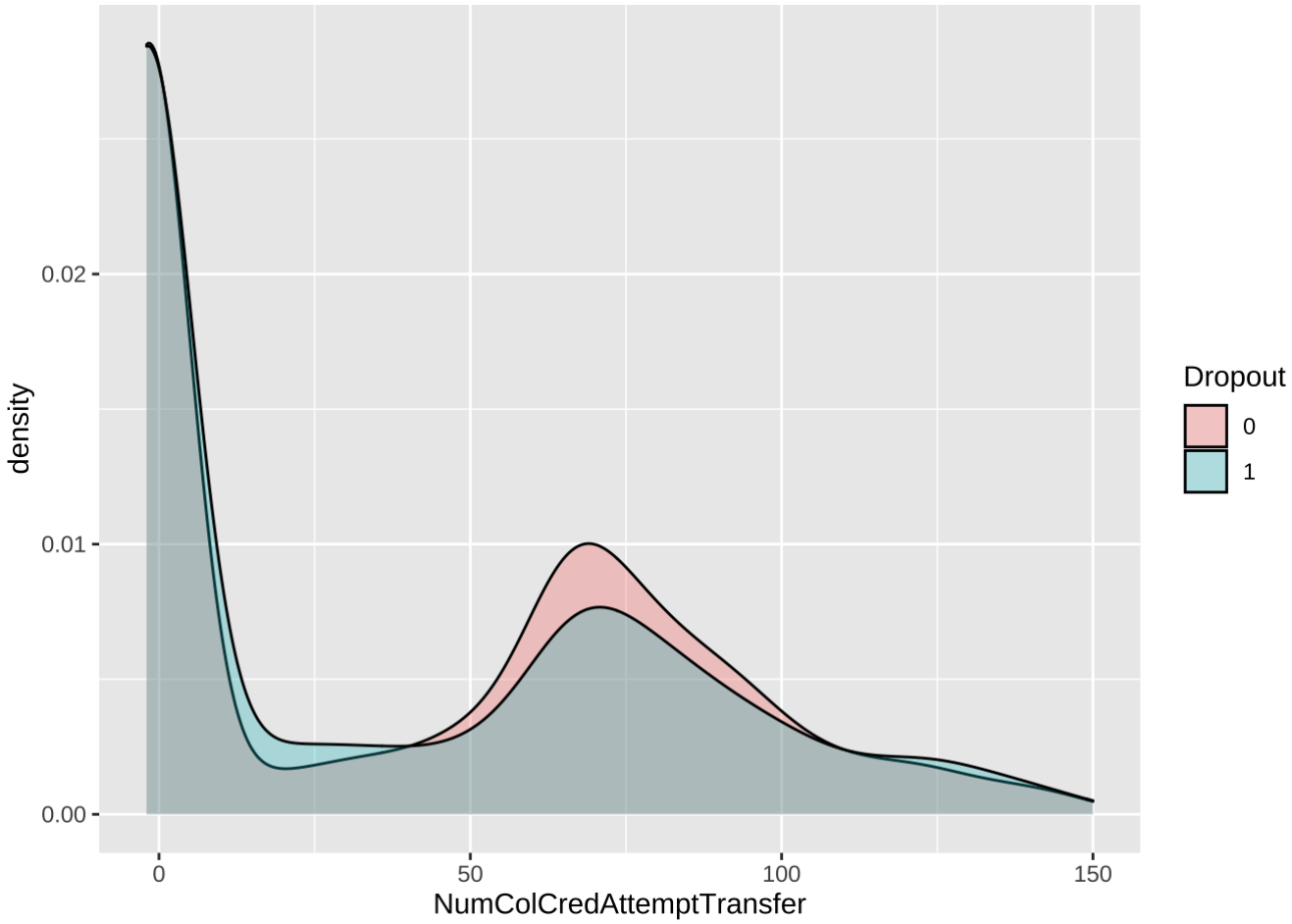



```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

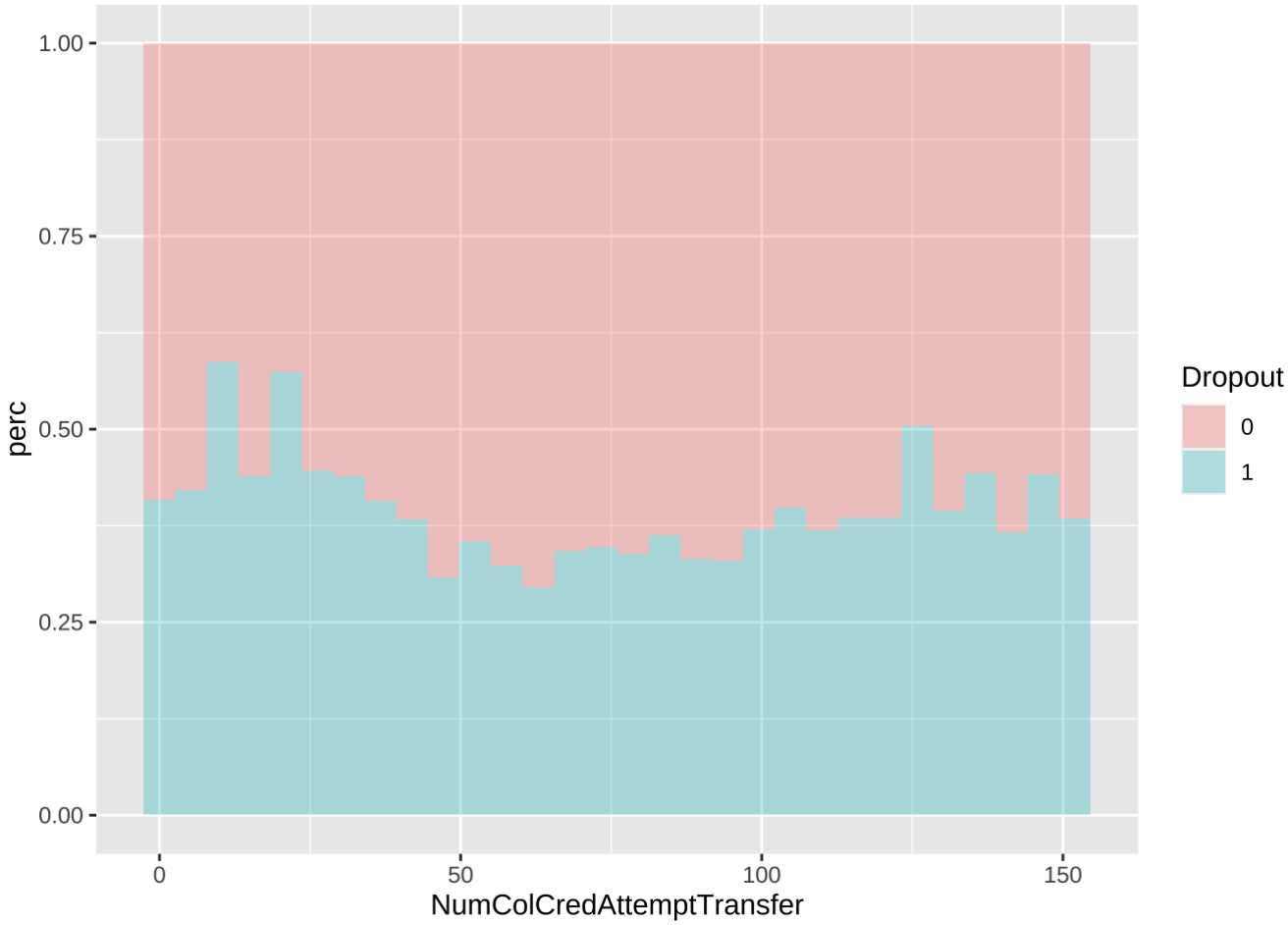
```
Warning: Removed 1 rows containing non-finite values (stat_bin).
```




```
Warning: Removed 1 rows containing non-finite values (stat_density).
```




```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

Warning: Removed 1 rows containing non-finite values (stat_bin).
```



Analysis and Main Results:

Nither NumColCredAcceptTransfer nor NumColCredAttemptTransfer is normally distributed, so Wilcoxon rank sum test was conducted and results show that they are significantly correlated with dropout.

So they will be involved and could be further selected in the model.

Involved variables:

- NumColCredAcceptTransfer
- NumColCredAttemptTransfer

Final Gpa and valid_transfer

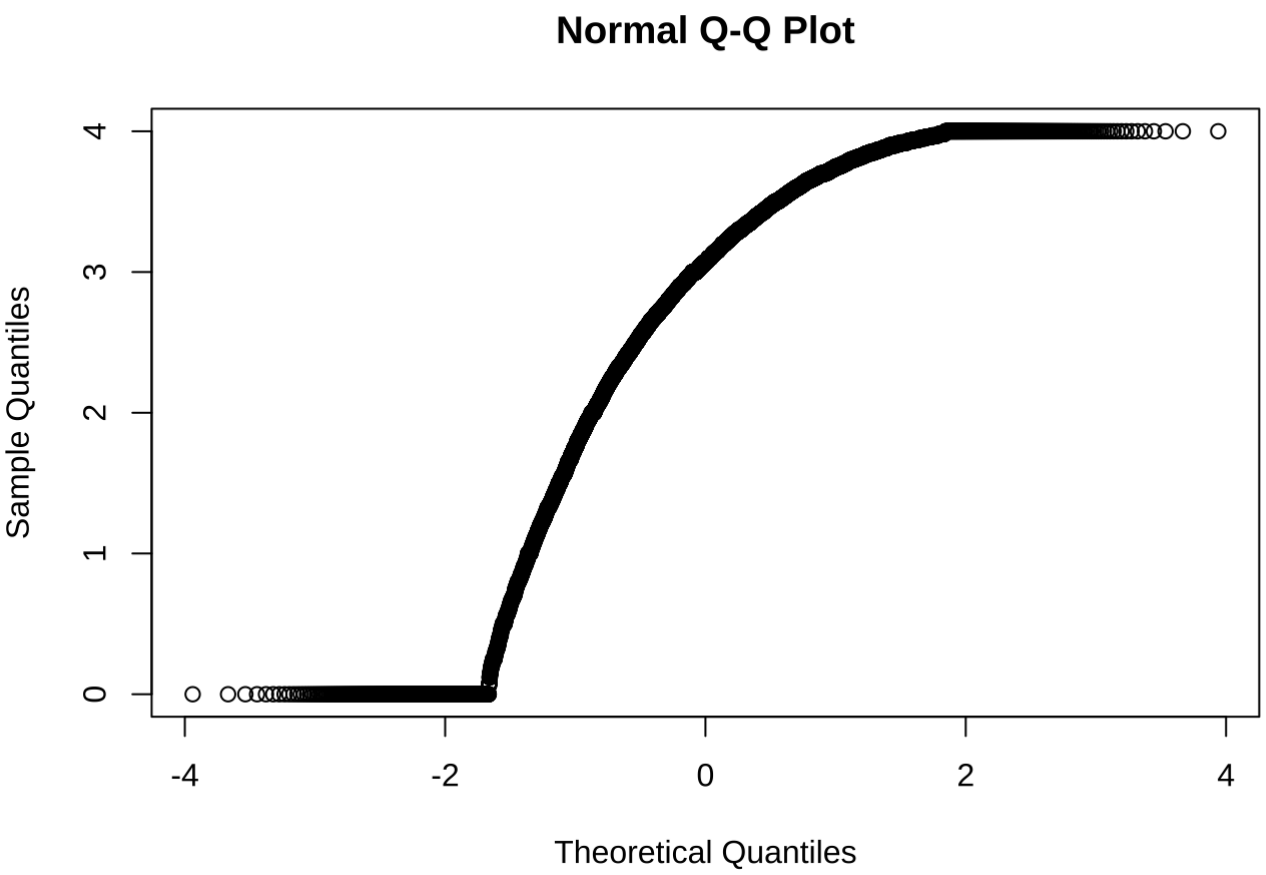
```
############  normal distribution test
ks.test(train_EDA$final_GPA,"pnorm")
```

```
Warning in ks.test.default(train_EDA$final_GPA, "pnorm"): ties should not be
present for the Kolmogorov-Smirnov test
```

```
    Asymptotic one-sample Kolmogorov-Smirnov test
```

```
data:  train_EDA$final_GPA
D = 0.8051, p-value < 2.2e-16
alternative hypothesis: two-sided
```

```
qqnorm(train_EDA$final_GPA)
```



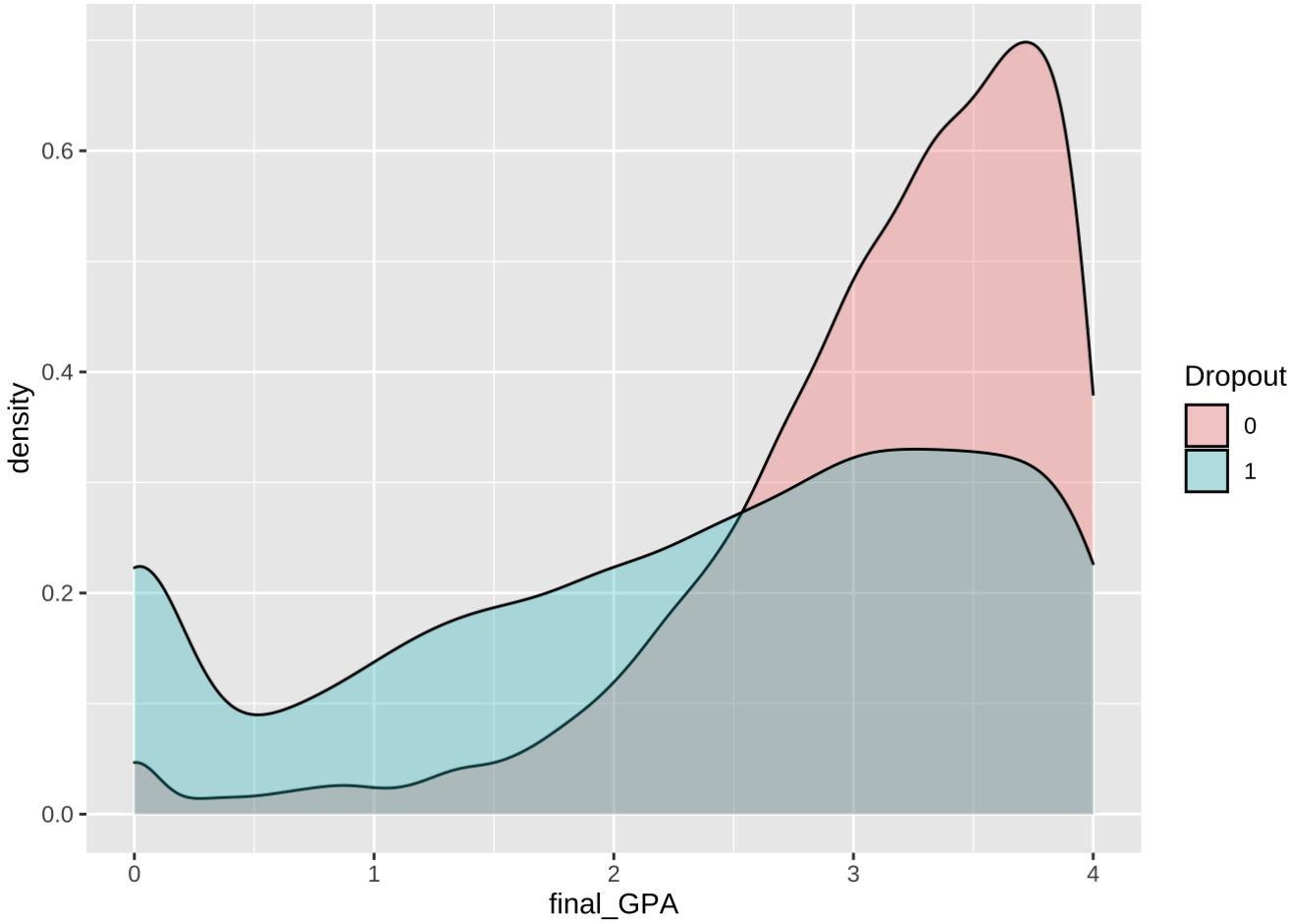

```
# ks.test(train_EDA$valid_transfer,"pnorm")
# qqnorm(train_EDA$valid_transfer)


####  Mann-Whitney U test
wilcox.test(train_EDA$final_GPA~train_EDA$Dropout)
```

```
    Wilcoxon rank sum test with continuity correction

data:  train_EDA$final_GPA by train_EDA$Dropout
W = 24484230, p-value < 2.2e-16
alternative hypothesis: true location shift is not equal to 0
```

```
# wilcox.test(train_EDA$valid_transfer~train_EDA$Dropout)
```

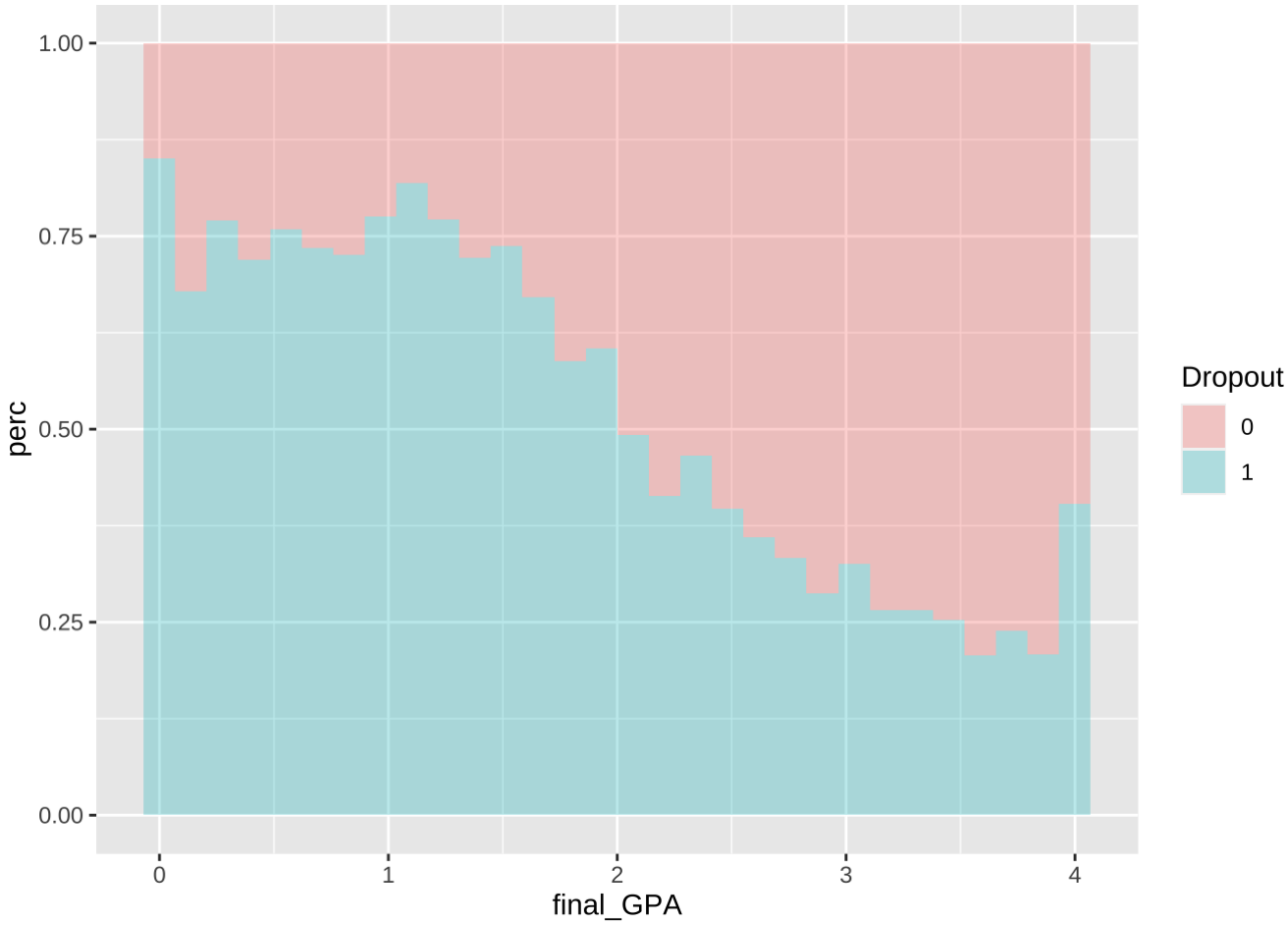
```
for(i in c("final_GPA")){
print(ggplot(train_EDA, aes(x = (get(i)), fill = Dropout)) +xlab(i)+geom_density(alpha =
print(ggplot(data = train_EDA, mapping = aes(x = get(i), fill =Dropout)) +xlab(i)+ ylab("
}
```

```
Warning: Removed 1 rows containing non-finite values (stat_density).
```




```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
Warning: Removed 1 rows containing non-finite values (stat_bin).
```



Analysis and Main Results:

Final Gpa is not normally distributed, so Wilcoxon rank sum test was conducted and result show that they Final Gpa is significantly correlated with dropout.

So this will be involved and could be further selected in the model.

Involved variables:

- Final_Gpa
- valid_transfer

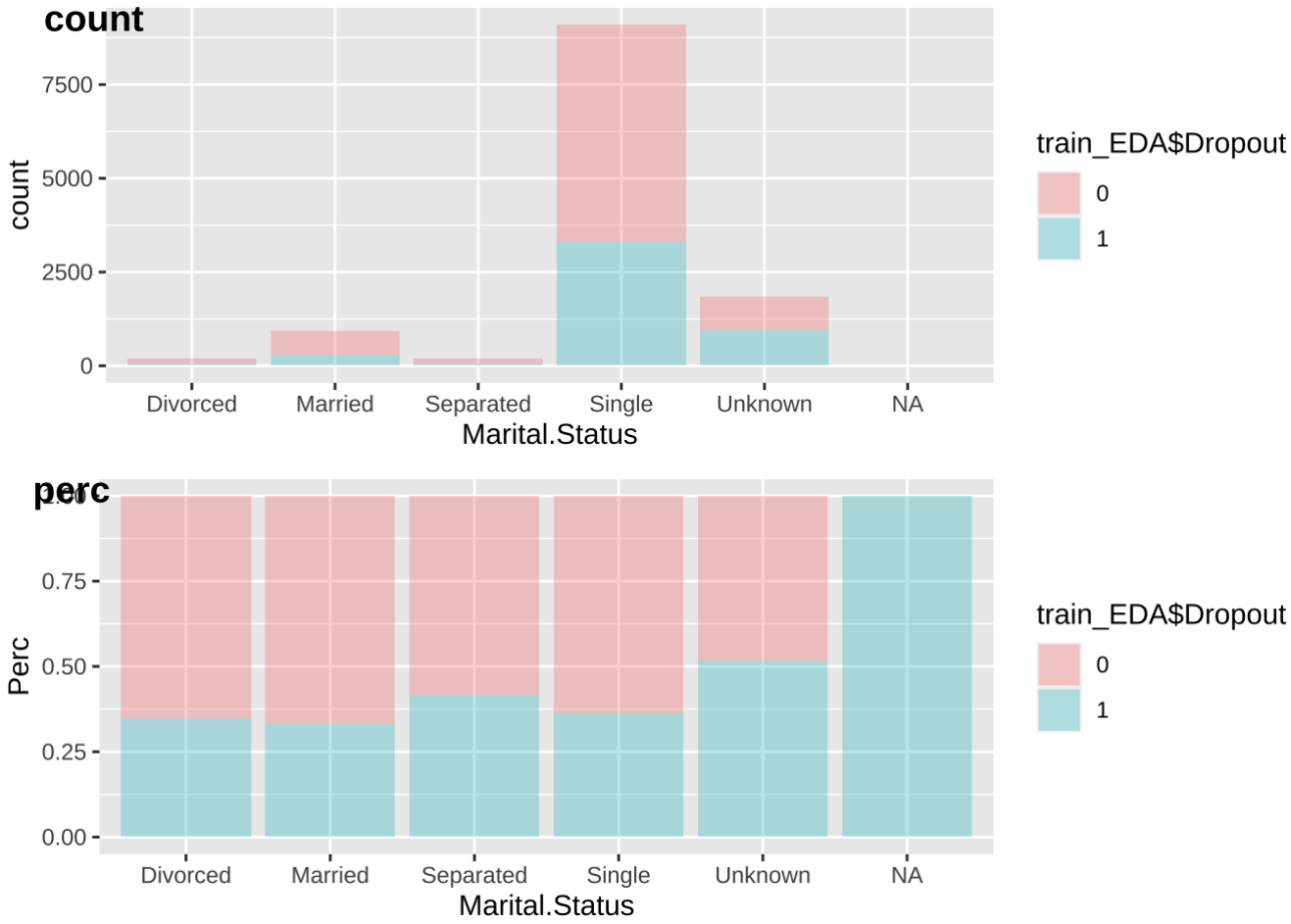
3.2.2 Interval & Ratio level variables（Discrete variable）:

Variables overview:

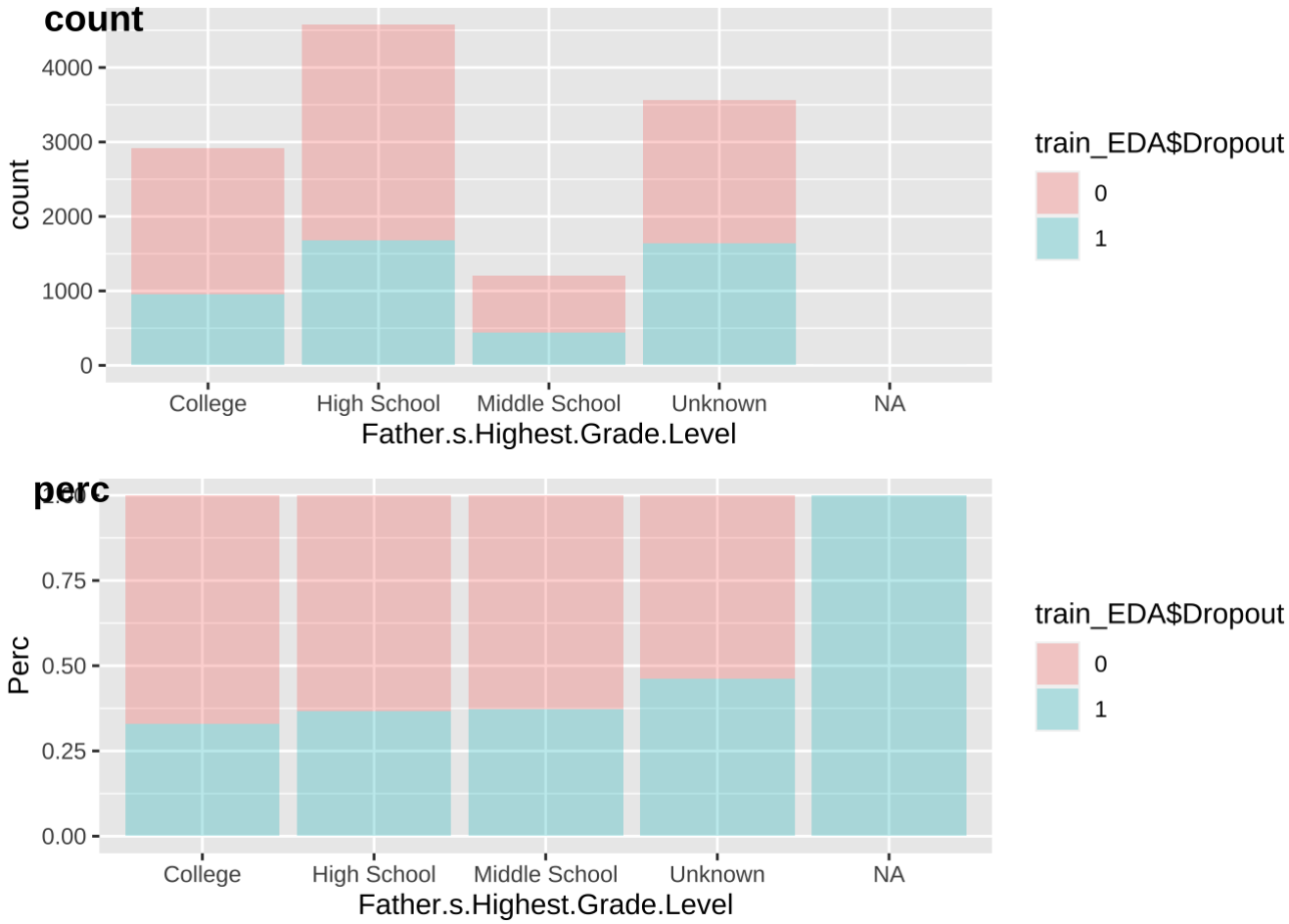
Finance	Static	Progress
Marital.Status	BirthMonth	CompleteDevMath /English
Father.s.Highest.Grade.Level	Gender	final_Complete1
Mother.s.Highest.Grade.Level	Race	first_term_Major1

Finance	Static	Progress
Housing	EnrollmentStatus	final_majorOne
	HighDeg	
	Math/EngPlacement	
	GatewayMath/EngStatus	

```
var <- list("Marital.Status", "Father.s.Highest.Grade.Level", "Mother.s.Highest.Grade.Lev

drop <- c()
for (i in var) {
  count <- ggplot(data.frame(train_EDA[, i]), aes(x = train_EDA[, i], fill = train_EDA$Dr
  perc <- ggplot(data.frame(train_EDA[, i]), aes(x = train_EDA[, i], fill = train_EDA$Dro
  print(ggarrange(count, perc, labels = c("count", "perc"), ncol = 1, nrow = 2))

  print(i)
  assign(i , with(train_EDA, table(Dropout, get(i))))
  table <- chisq.test(get(i))
  print(chisq.test(get(i)))

  if(is.na(table$p.value)) {
    next
  }
  if(table$p.value >0.001) {
    drop <- append(drop, i)
  }
}
```

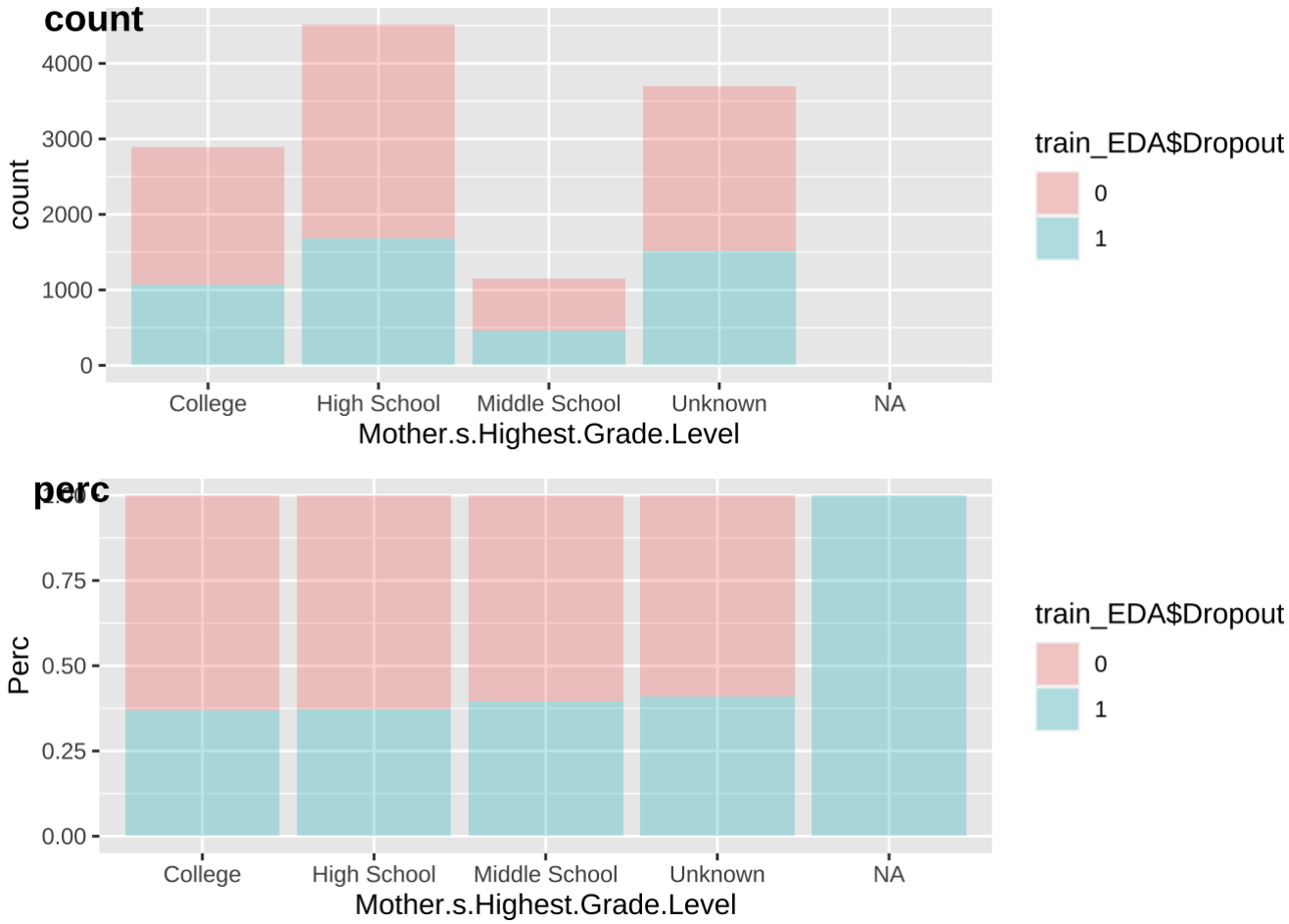



```
[1] "Marital.Status"

    Pearson's Chi-squared test

data:  get(i)
X-squared = 161.07, df = 4, p-value < 2.2e-16
```

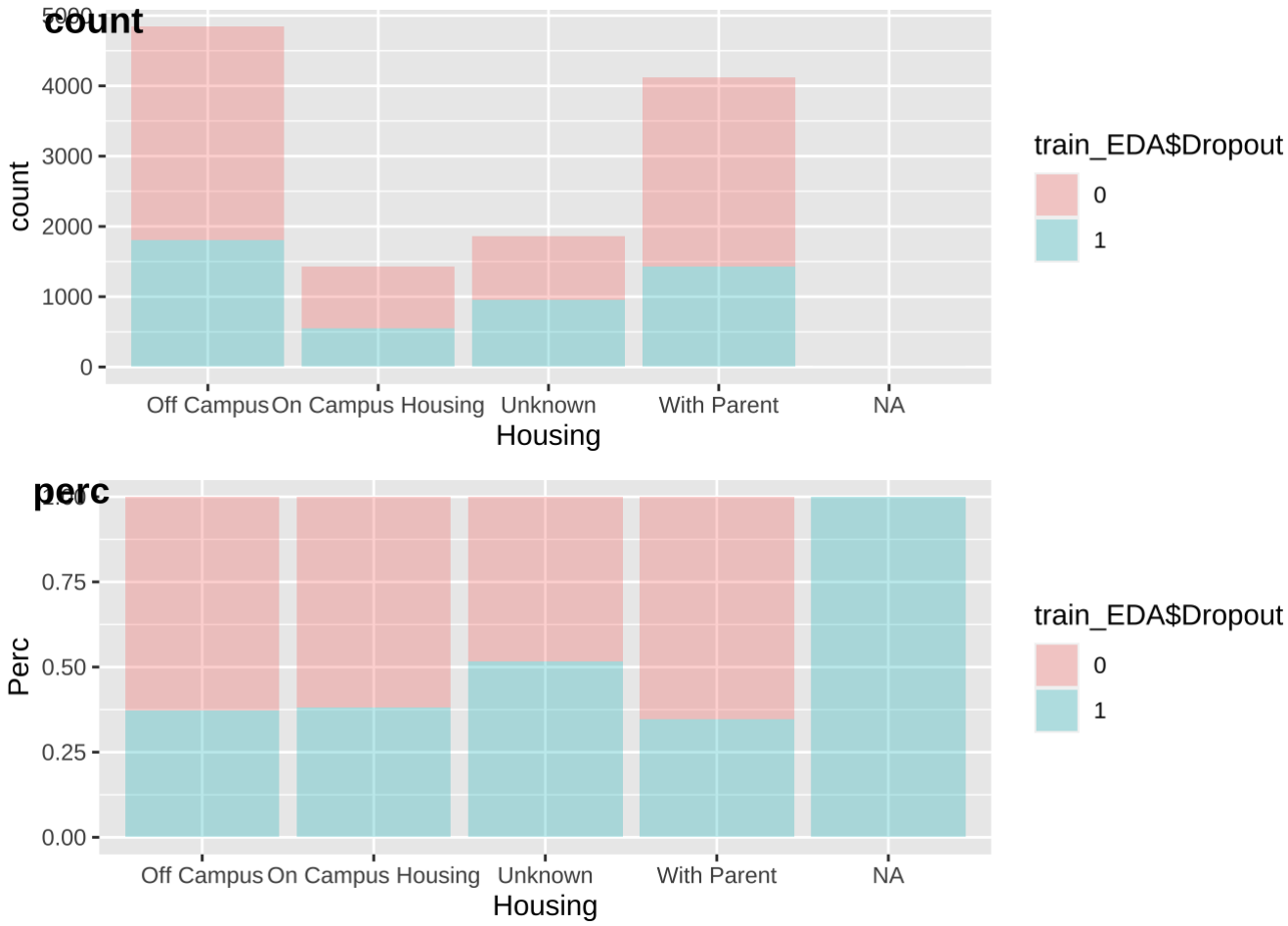



```
[1] "Father.s.Highest.Grade.Level"
```

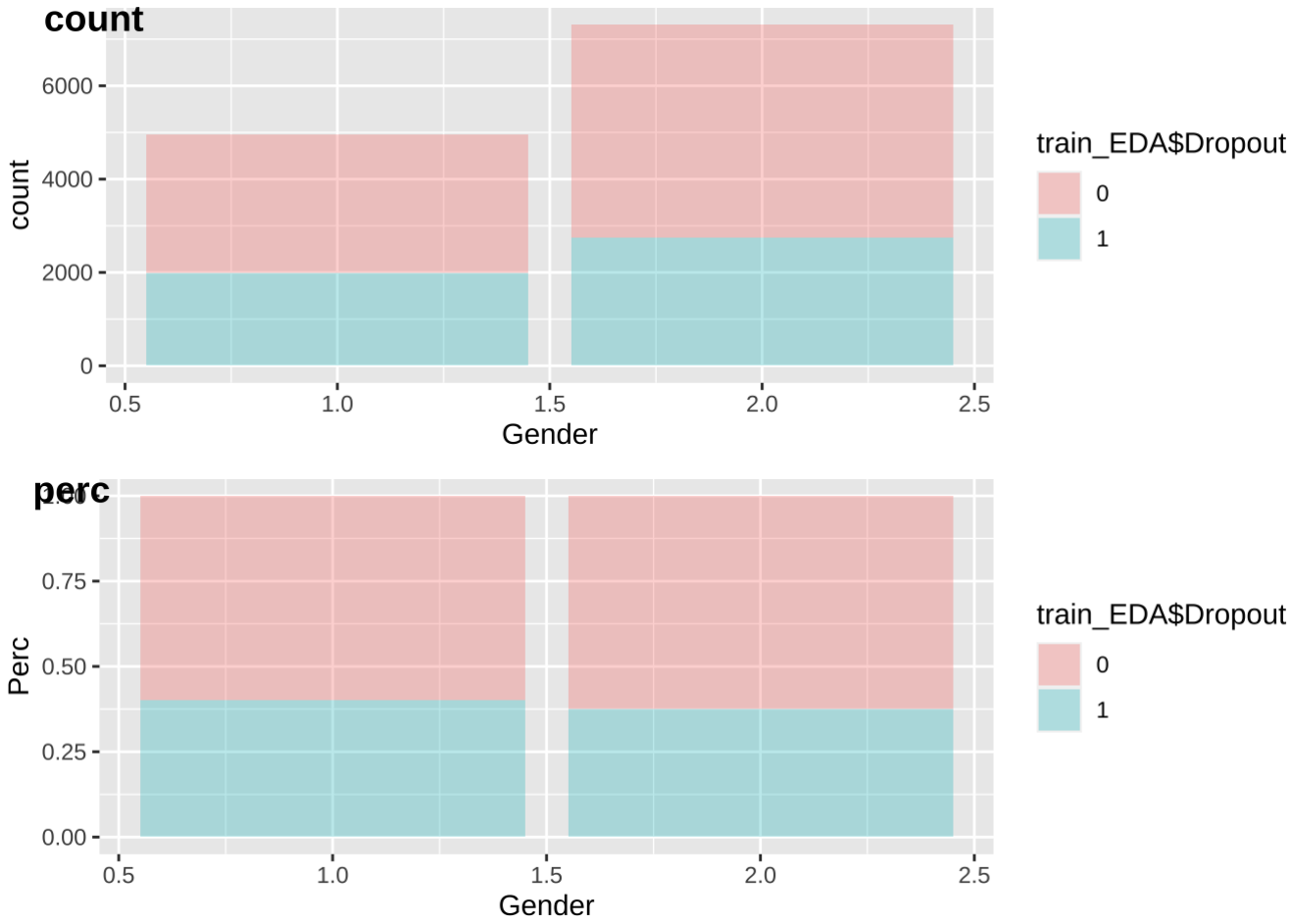
```
    Pearson's Chi-squared test

data:  get(i)
X-squared = 134, df = 3, p-value < 2.2e-16
```



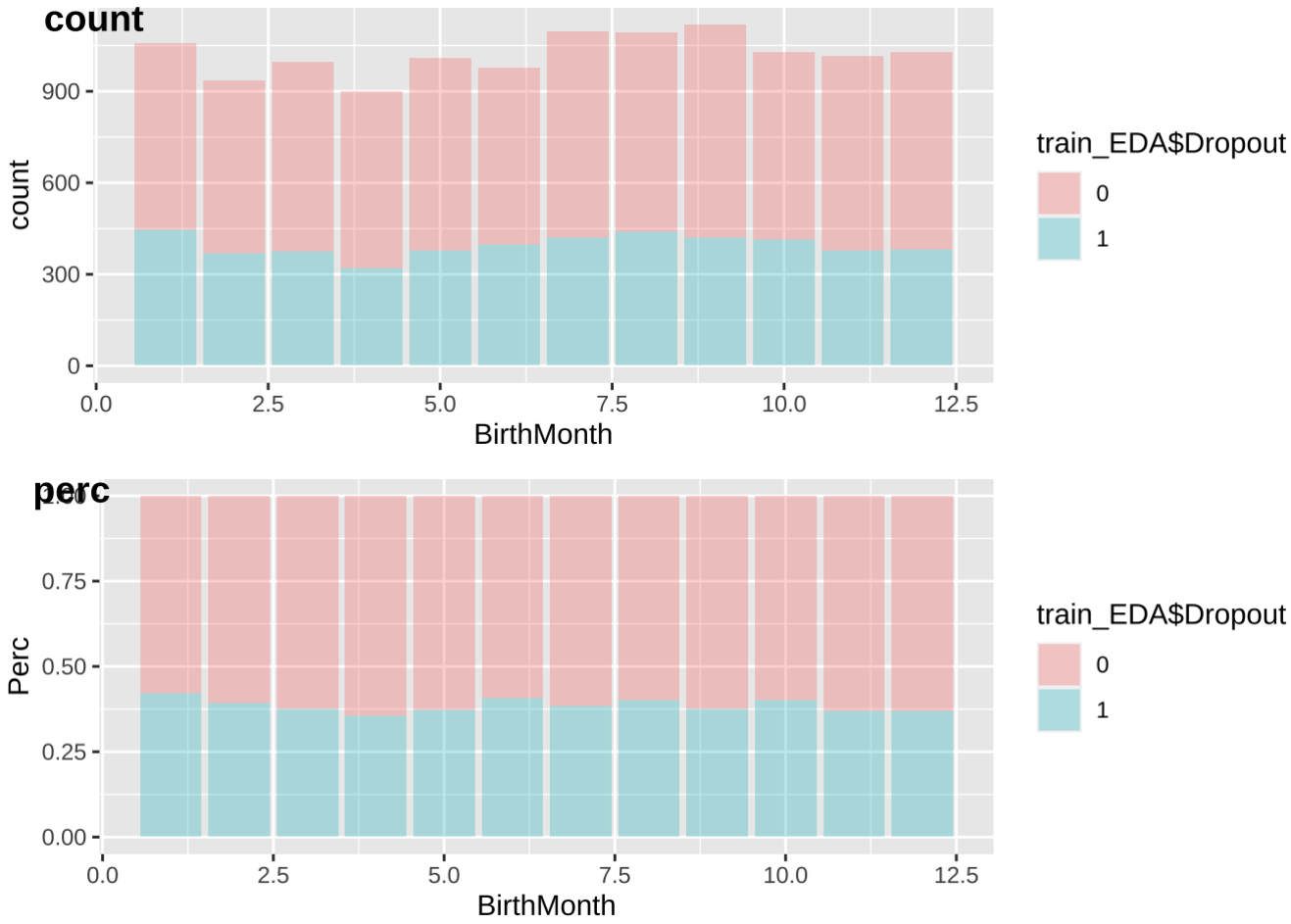

```
[1] "Mother.s.Highest.Grade.Level"

    Pearson's Chi-squared test

data:  get(i)
X-squared = 15.567, df = 3, p-value = 0.001391

[1] "Housing"

    Pearson's Chi-squared test

data:  get(i)
X-squared = 165.97, df = 3, p-value < 2.2e-16

Warning: Removed 1 rows containing non-finite values (stat_count).
Removed 1 rows containing non-finite values (stat_count).
```



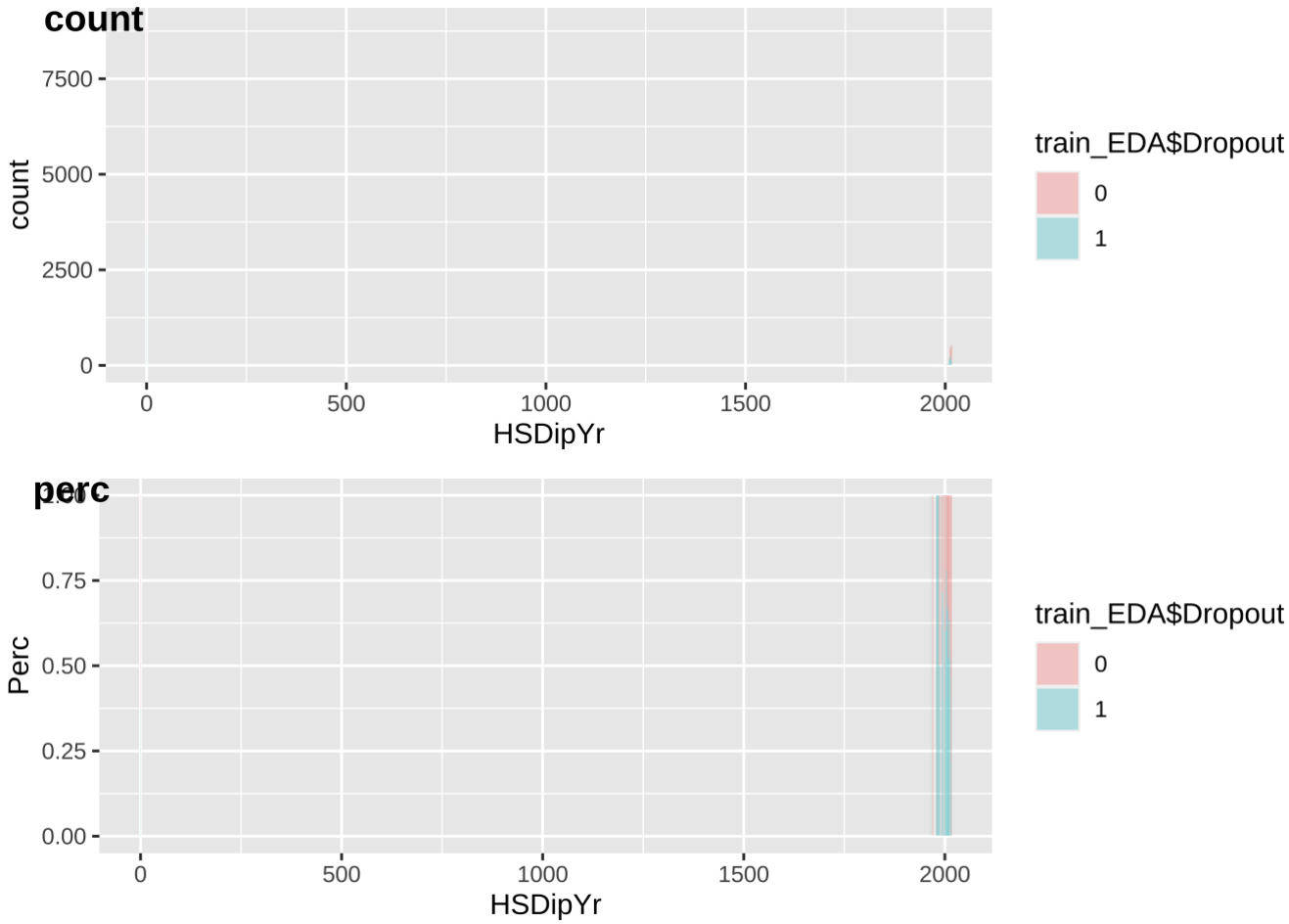

```
[1] "Gender"

    Pearson's Chi-squared test with Yates' continuity correction

data:  get(i)
X-squared = 8.8557, df = 1, p-value = 0.002922

Warning: Removed 1 rows containing non-finite values (stat_count).
Removed 1 rows containing non-finite values (stat_count).
```



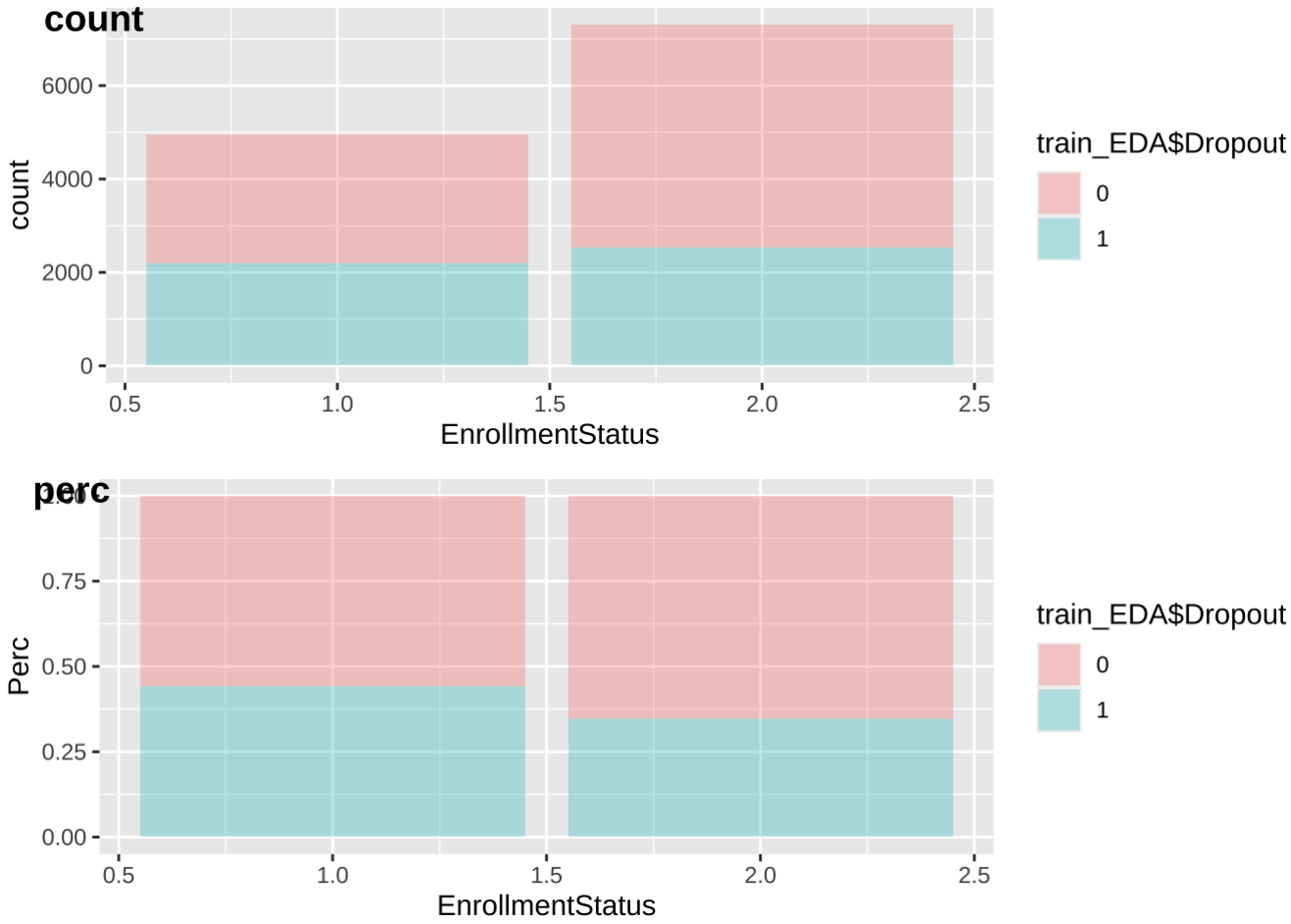

```
[1] "BirthMonth"

    Pearson's Chi-squared test

data:  get(i)
X-squared = 17.288, df = 11, p-value = 0.09965

Warning: Removed 1 rows containing non-finite values (stat_count).
Removed 1 rows containing non-finite values (stat_count).
```



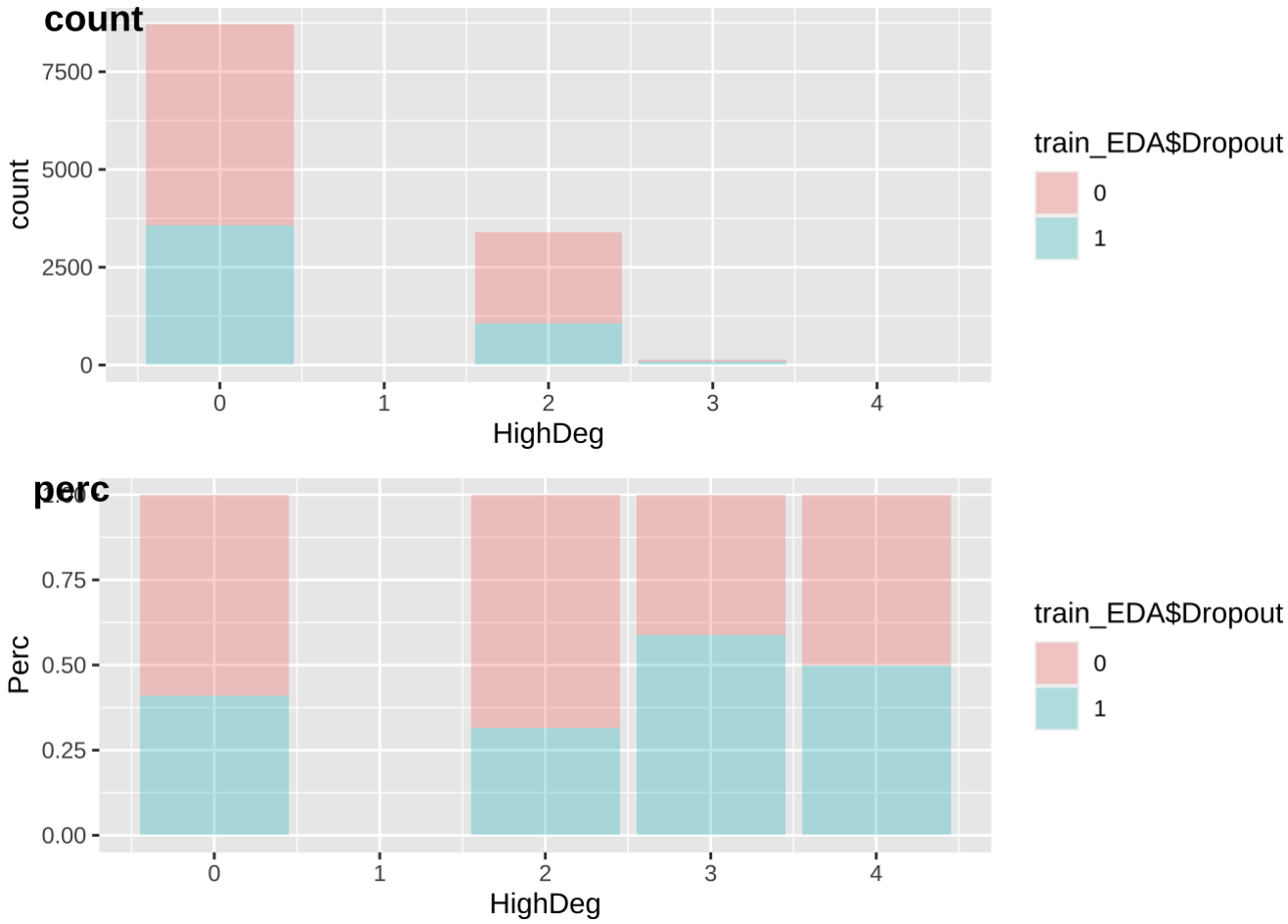

```
[1] "HSDipYr"

Warning in chisq.test(get(i)): Chi-squared approximation may be incorrect

Warning in chisq.test(get(i)): Chi-squared approximation may be incorrect


    Pearson's Chi-squared test

data:  get(i)
X-squared = 786.78, df = 36, p-value < 2.2e-16

Warning: Removed 1 rows containing non-finite values (stat_count).

Warning: Removed 1 rows containing non-finite values (stat_count).
```



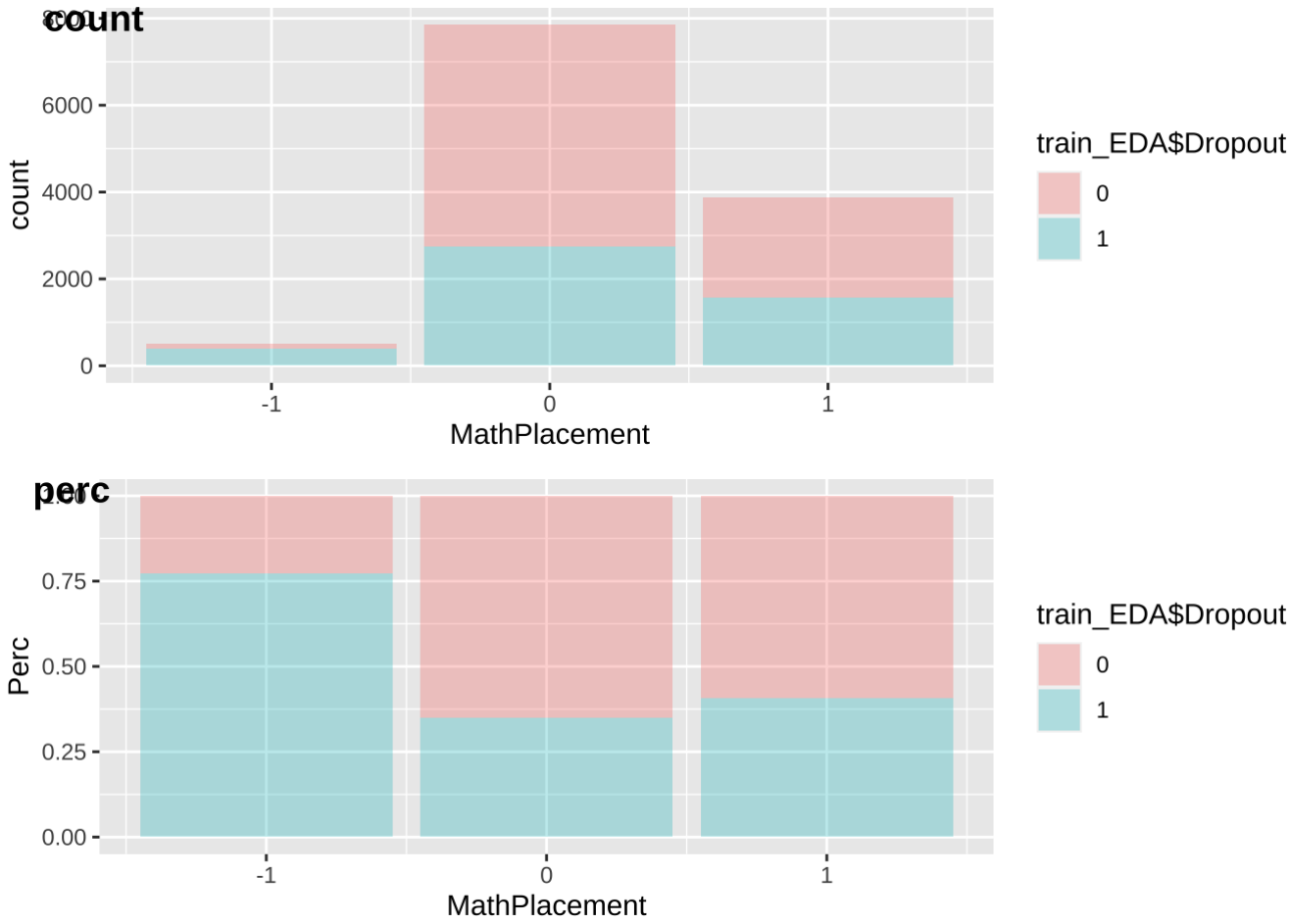

```
[1] "EnrollmentStatus"

    Pearson's Chi-squared test with Yates' continuity correction

data:  get(i)
X-squared = 111.88, df = 1, p-value < 2.2e-16

Warning: Removed 1 rows containing non-finite values (stat_count).
Removed 1 rows containing non-finite values (stat_count).
```



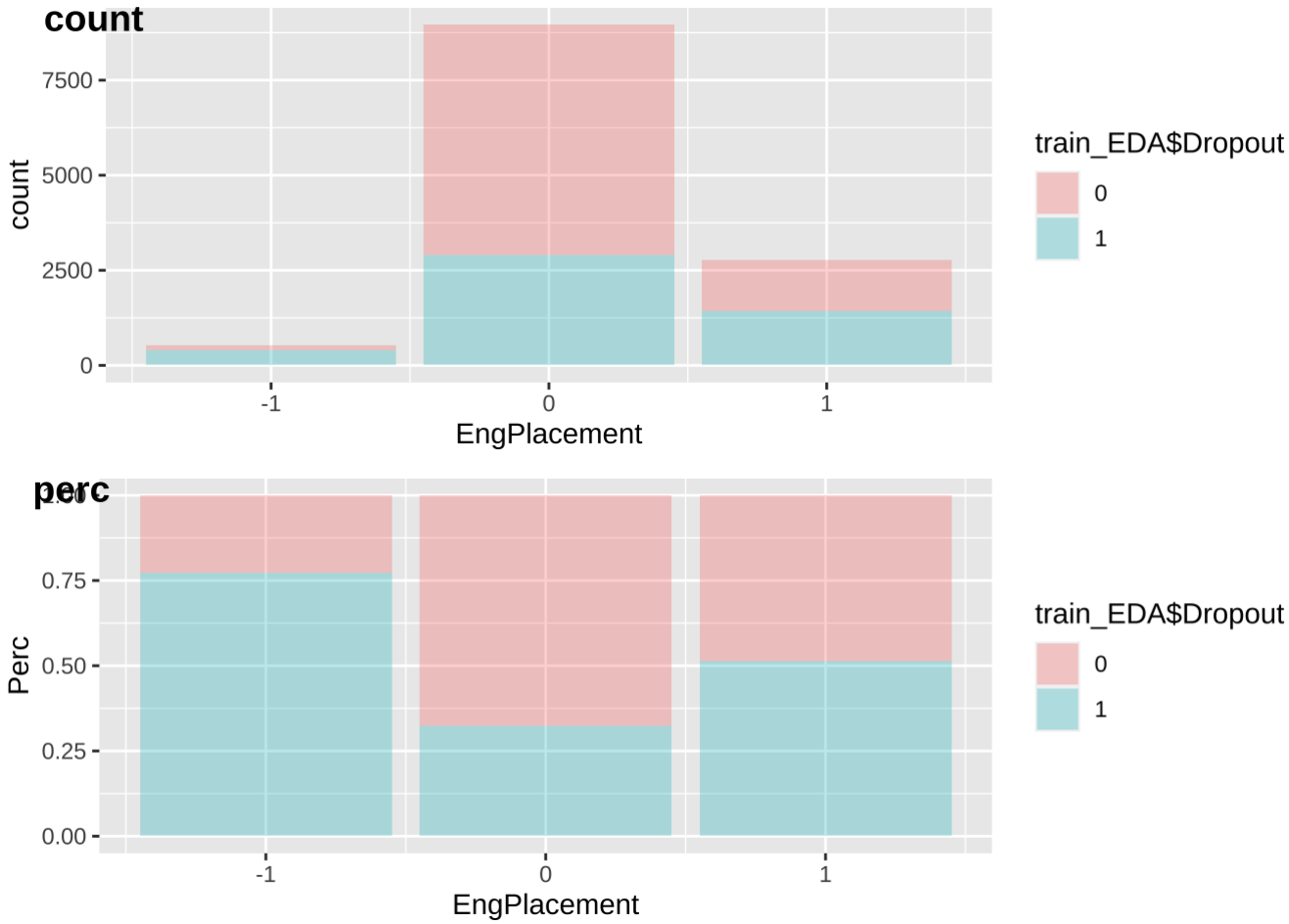

```
[1] "HighDeg"

Warning in chisq.test(get(i)): Chi-squared approximation may be incorrect

Warning in chisq.test(get(i)): Chi-squared approximation may be incorrect


    Pearson's Chi-squared test

data:  get(i)
X-squared = 120.62, df = 3, p-value < 2.2e-16

Warning: Removed 1 rows containing non-finite values (stat_count).

Warning: Removed 1 rows containing non-finite values (stat_count).
```



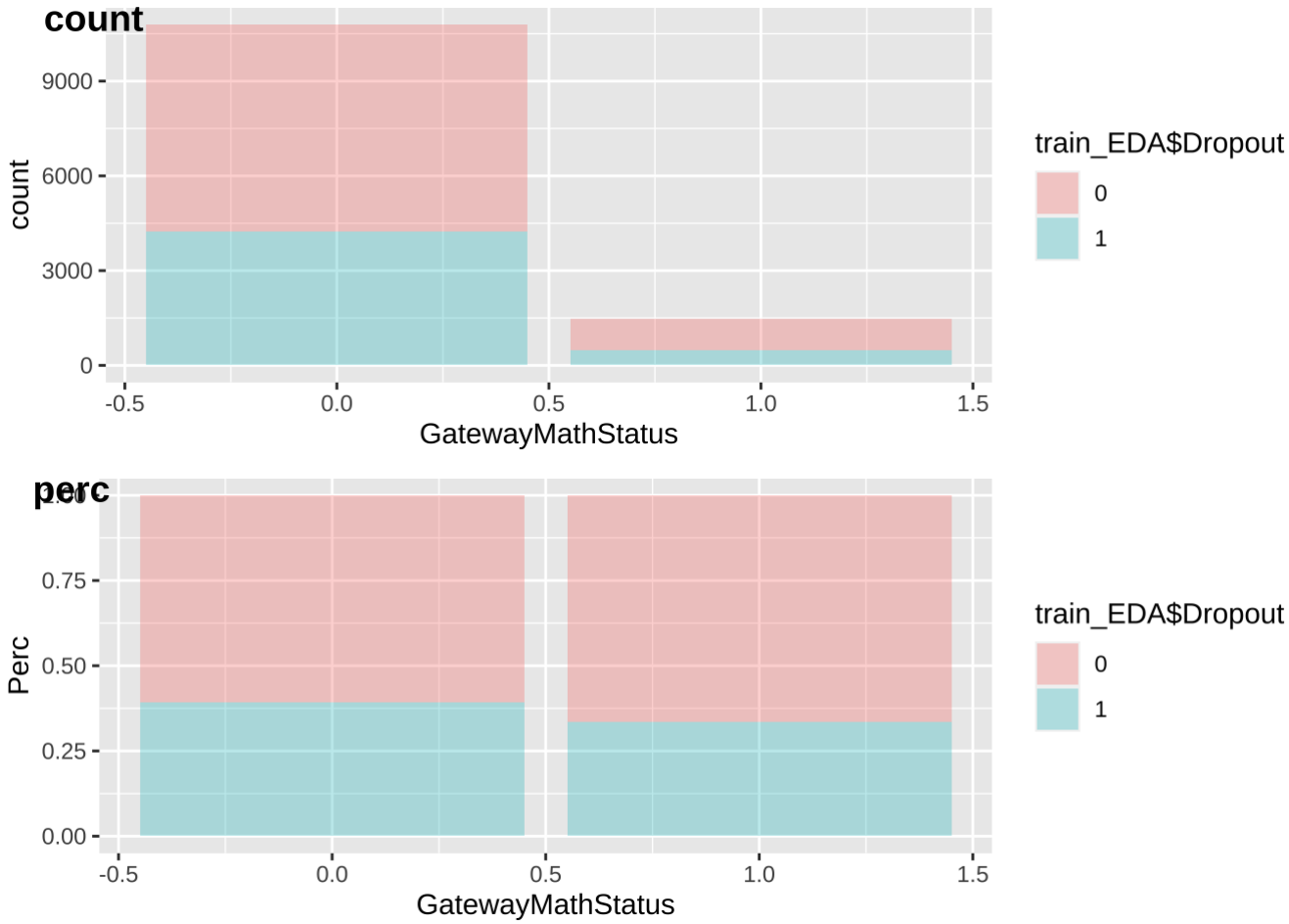

```
[1] "MathPlacement"

    Pearson's Chi-squared test

data:  get(i)
X-squared = 377.27, df = 2, p-value < 2.2e-16

Warning: Removed 1 rows containing non-finite values (stat_count).
Removed 1 rows containing non-finite values (stat_count).
```



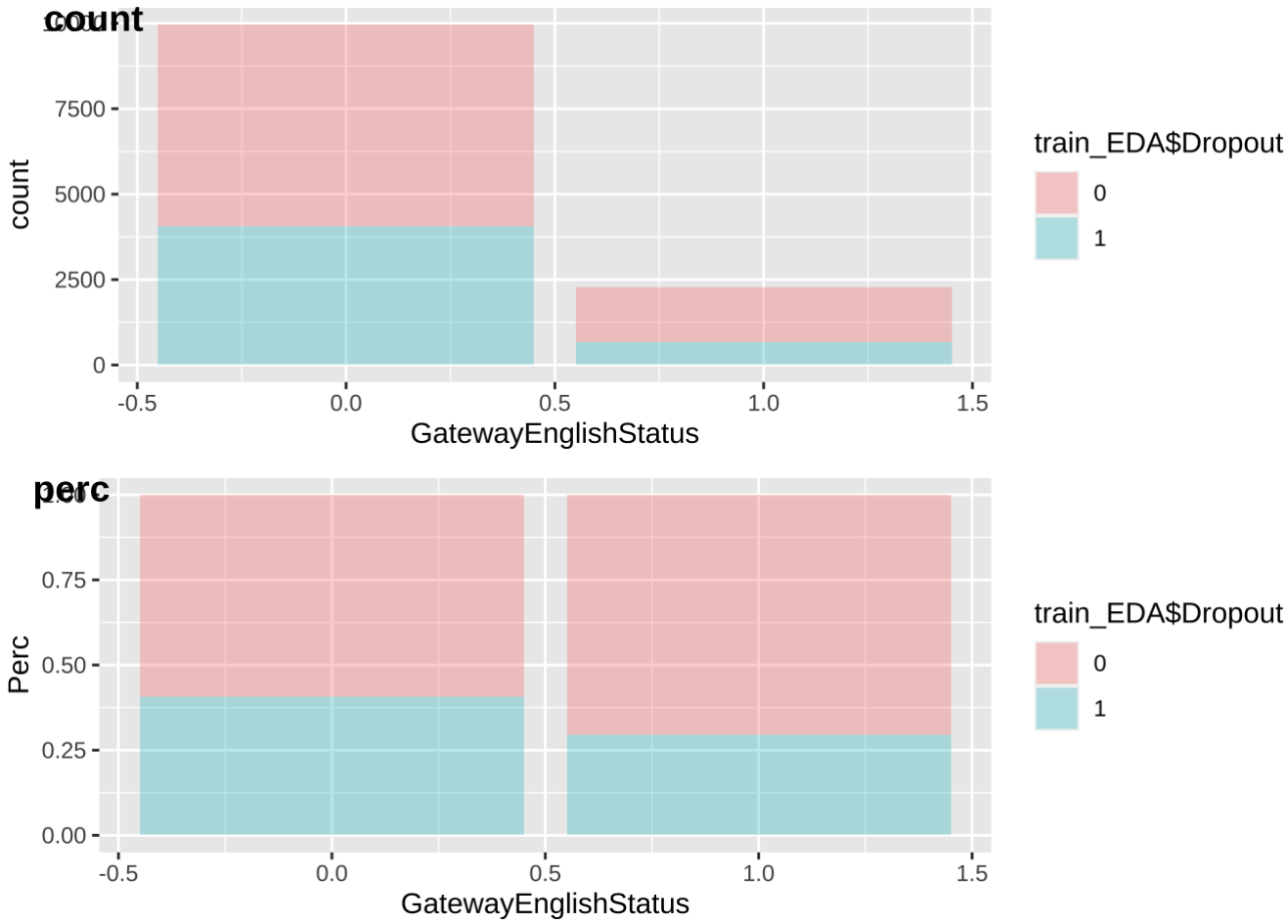

```
[1] "EngPlacement"

    Pearson's Chi-squared test

data:  get(i)
X-squared = 670.21, df = 2, p-value < 2.2e-16

Warning: Removed 1 rows containing non-finite values (stat_count).
Removed 1 rows containing non-finite values (stat_count).
```




```
[1] "GatewayMathStatus"

    Pearson's Chi-squared test with Yates' continuity correction

data:  get(i)
X-squared = 18.298, df = 1, p-value = 1.89e-05

Warning: Removed 1 rows containing non-finite values (stat_count).
Removed 1 rows containing non-finite values (stat_count).
```




```
[1] "GatewayEnglishStatus"

    Pearson's Chi-squared test with Yates' continuity correction

data:  get(i)
X-squared = 98.929, df = 1, p-value < 2.2e-16

Warning: Removed 1 rows containing non-finite values (stat_count).
Removed 1 rows containing non-finite values (stat_count).
```

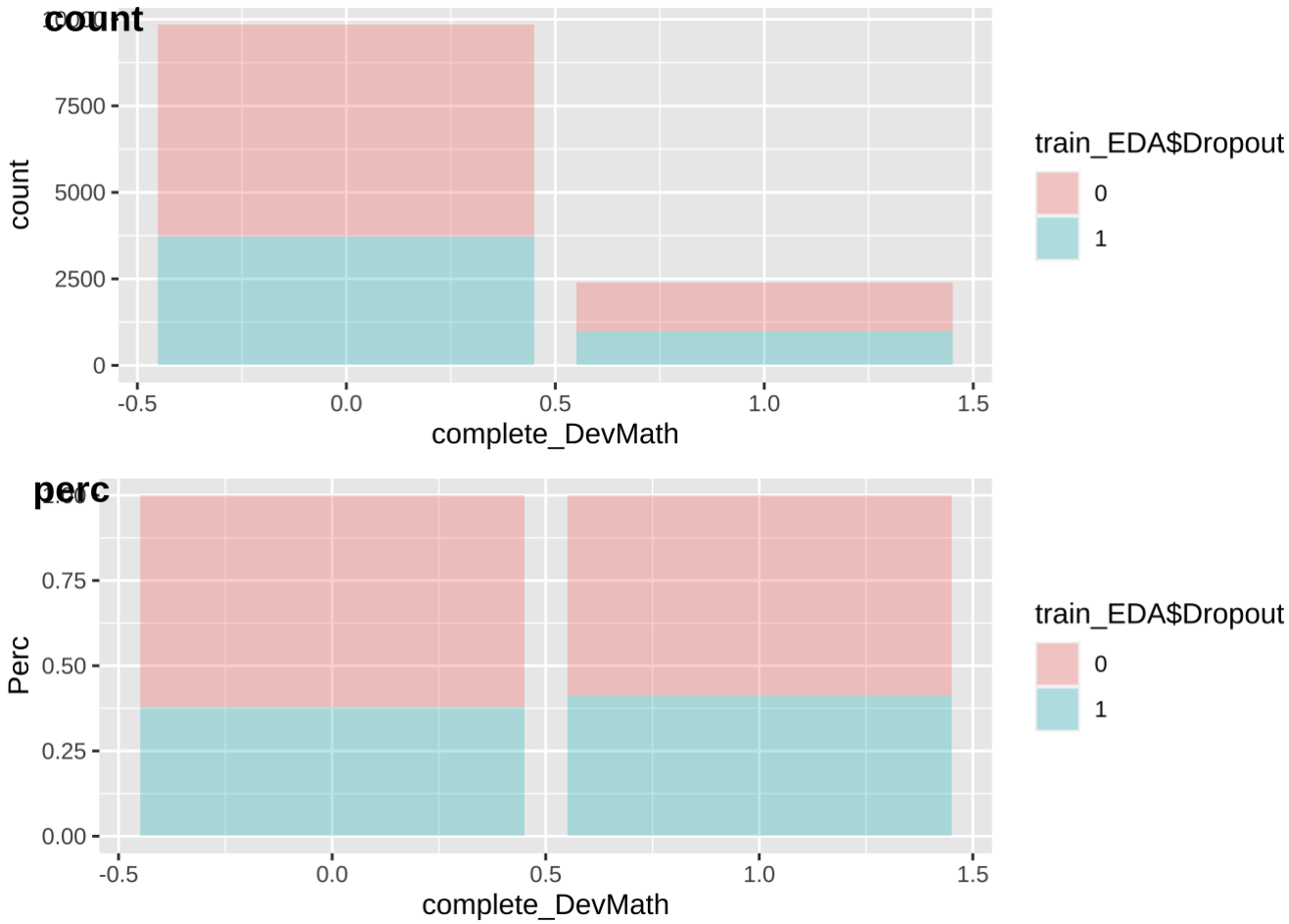



```
[1] "complete_DevMath"
```

```
    Pearson's Chi-squared test with Yates' continuity correction
```

```
data:  get(i)
X-squared = 8.2442, df = 1, p-value = 0.004088
```

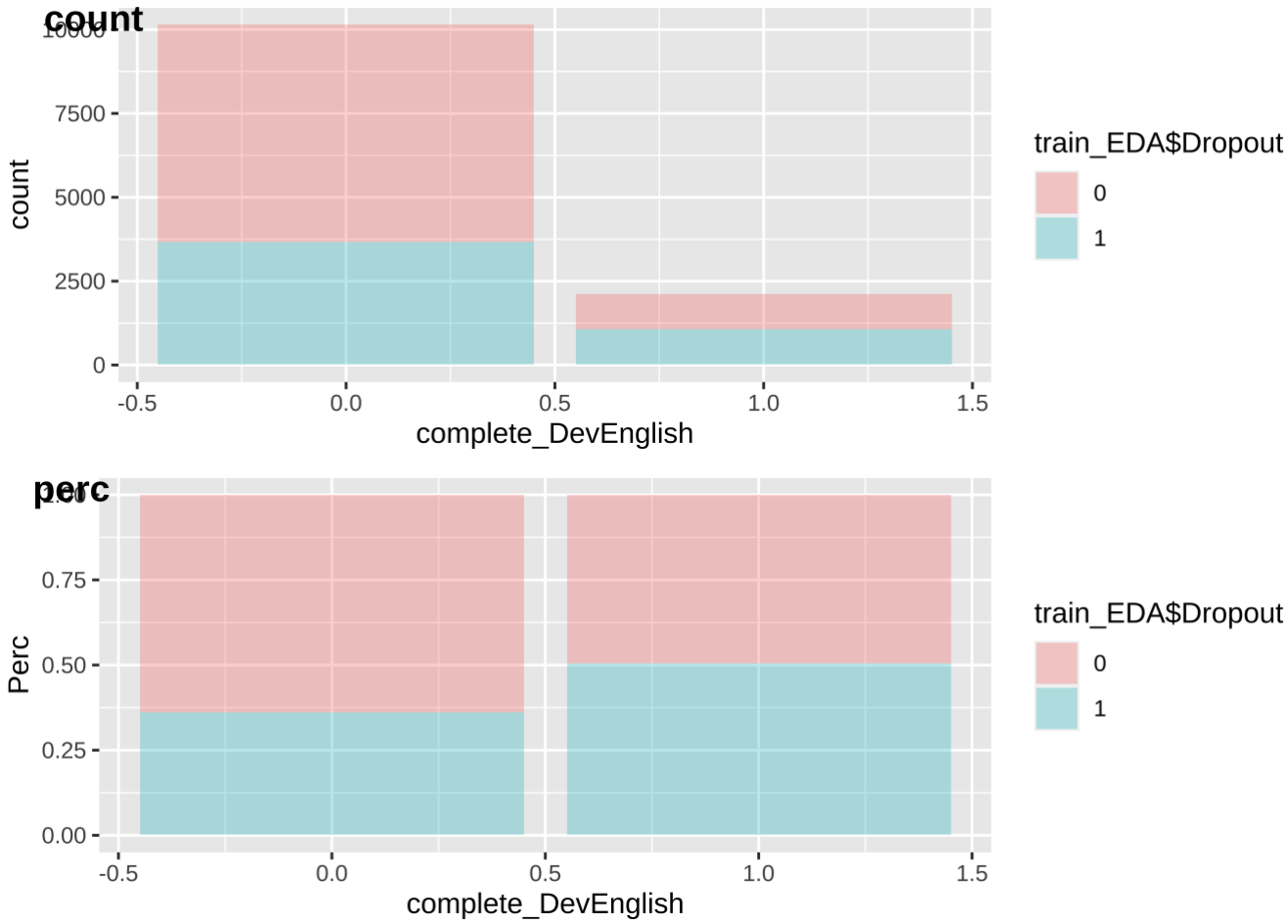
```
Warning: Removed 1 rows containing non-finite values (stat_count).
Removed 1 rows containing non-finite values (stat_count).
```



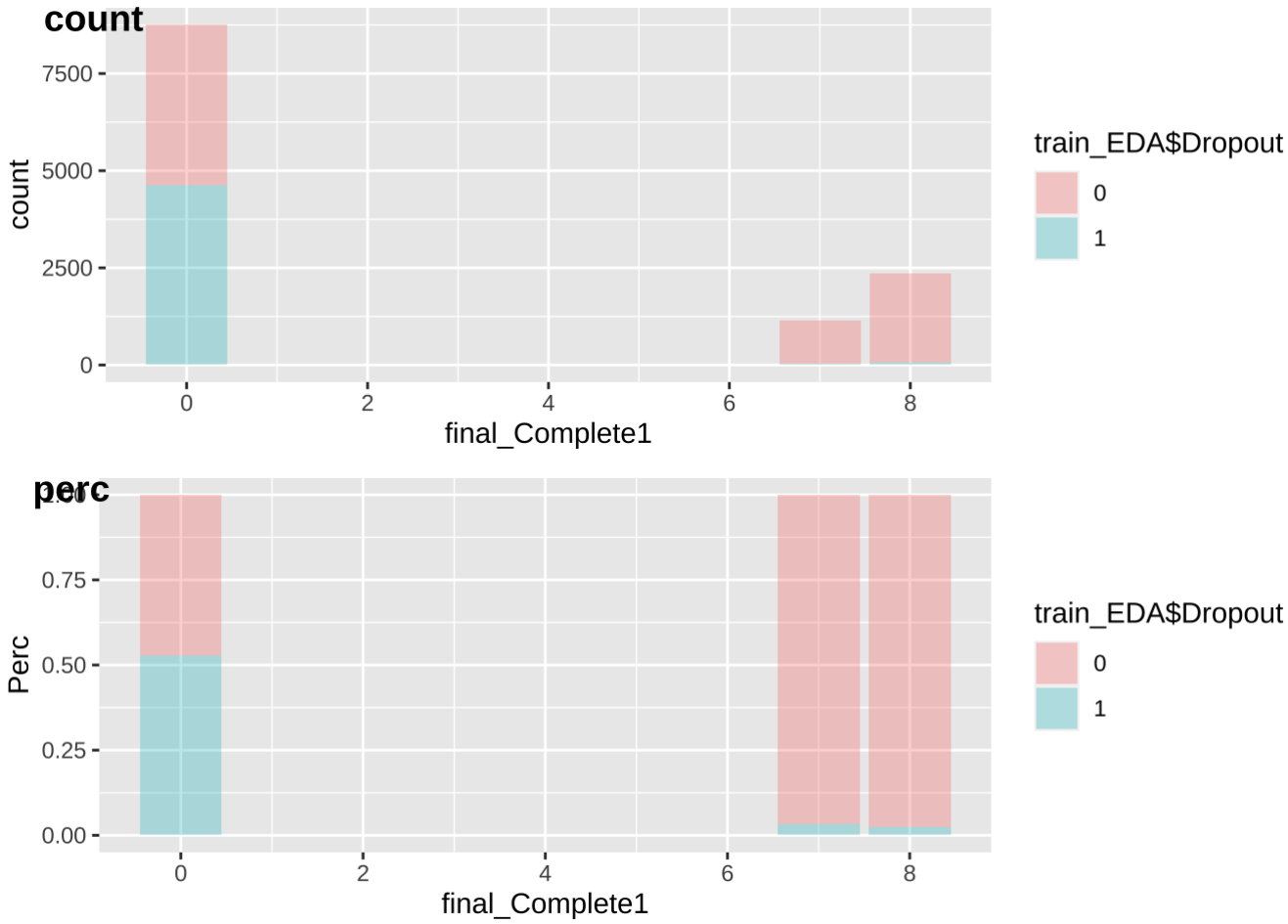

```
[1] "complete_DevEnglish"
```

```
    Pearson's Chi-squared test with Yates' continuity correction

data:  get(i)
X-squared = 154.35, df = 1, p-value < 2.2e-16
```

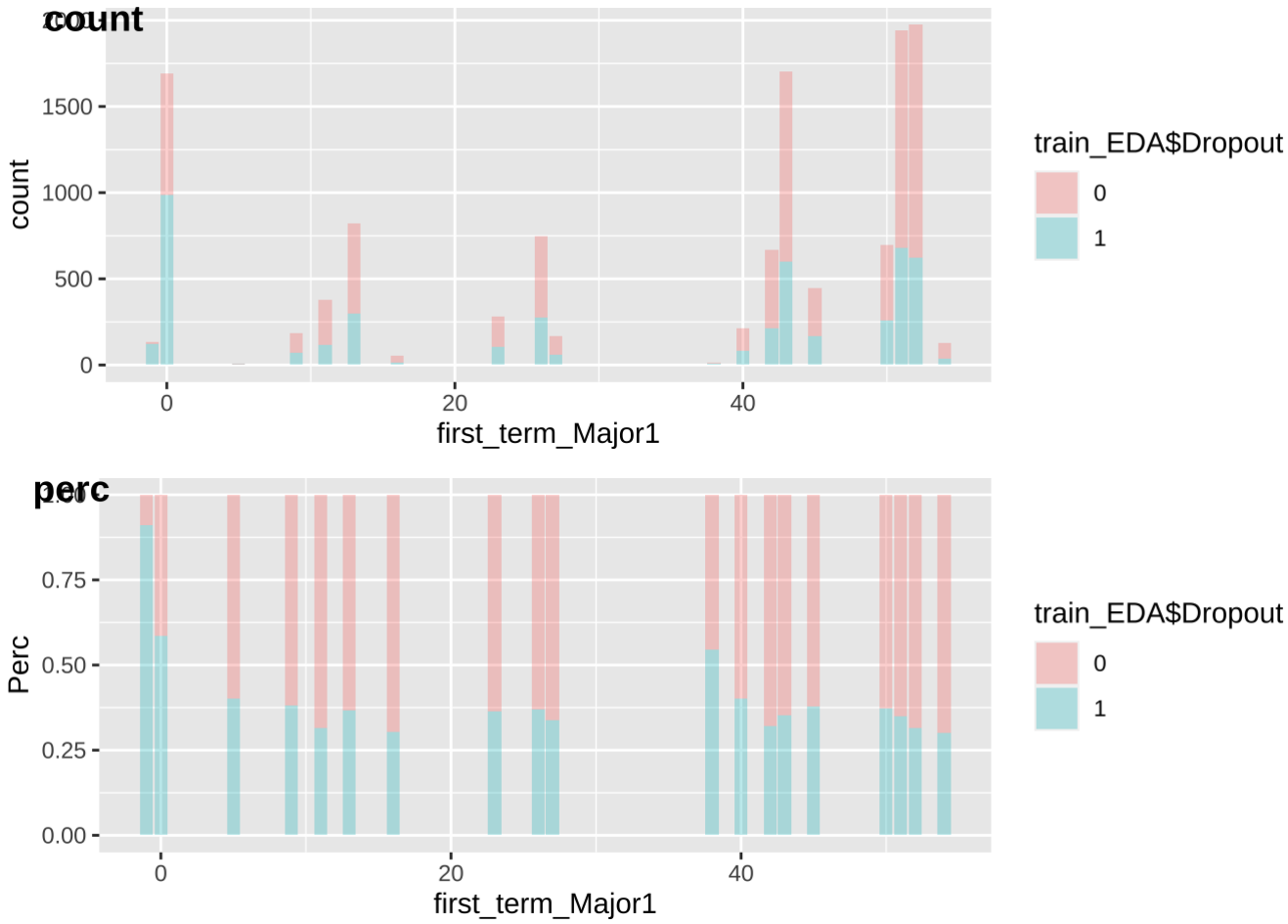
```
Warning: Removed 1 rows containing non-finite values (stat_count).
Removed 1 rows containing non-finite values (stat_count).
```




```
[1] "final_Complete1"

    Pearson's Chi-squared test

data:  get(i)
X-squared = 2650.5, df = 2, p-value < 2.2e-16

Warning: Removed 1 rows containing non-finite values (stat_count).
Removed 1 rows containing non-finite values (stat_count).
```



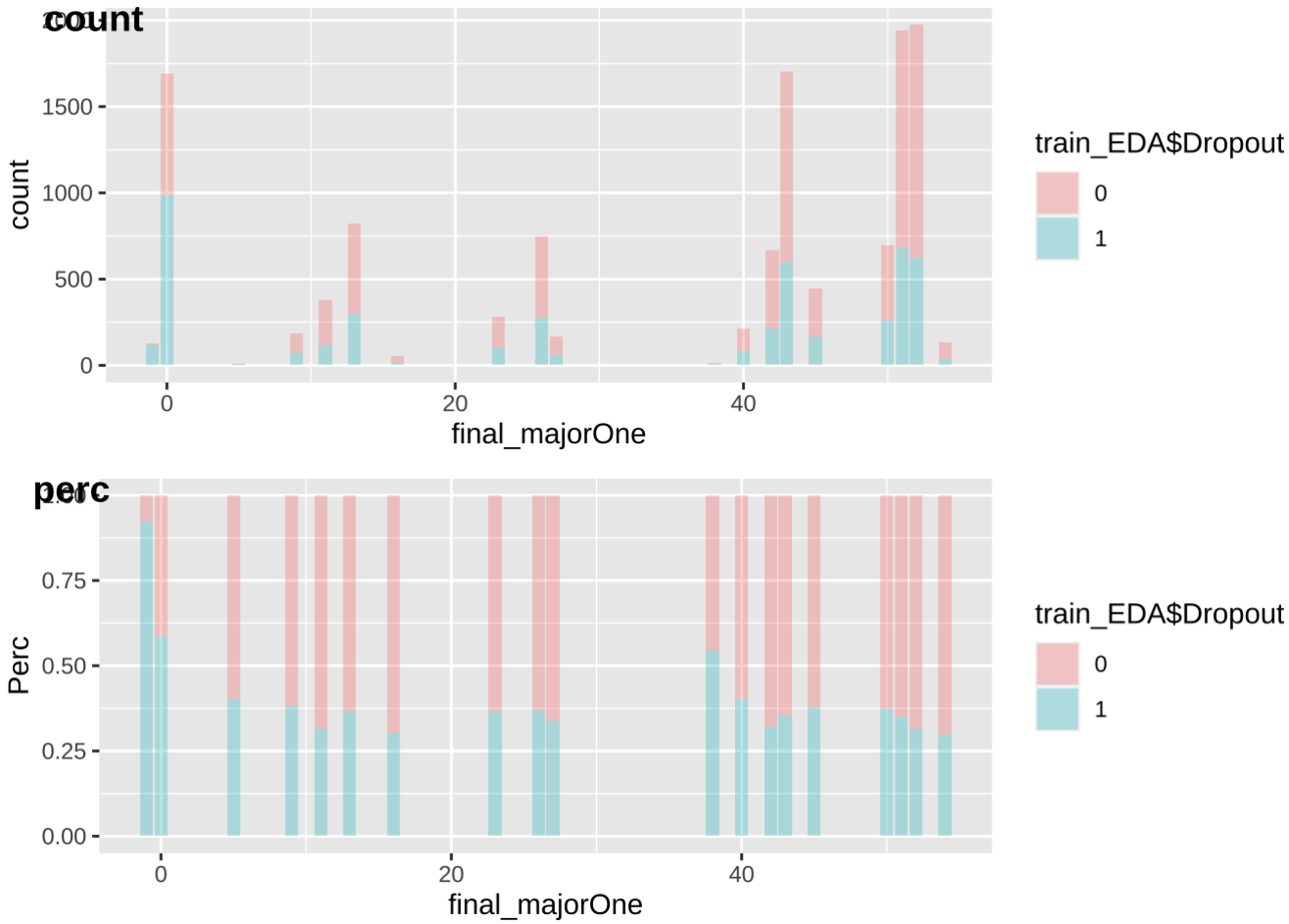

```
[1] "first_term_Major1"

Warning in chisq.test(get(i)): Chi-squared approximation may be incorrect

Warning in chisq.test(get(i)): Chi-squared approximation may be incorrect


    Pearson's Chi-squared test

data:  get(i)
X-squared = 526.34, df = 18, p-value < 2.2e-16

Warning: Removed 1 rows containing non-finite values (stat_count).

Warning: Removed 1 rows containing non-finite values (stat_count).
```



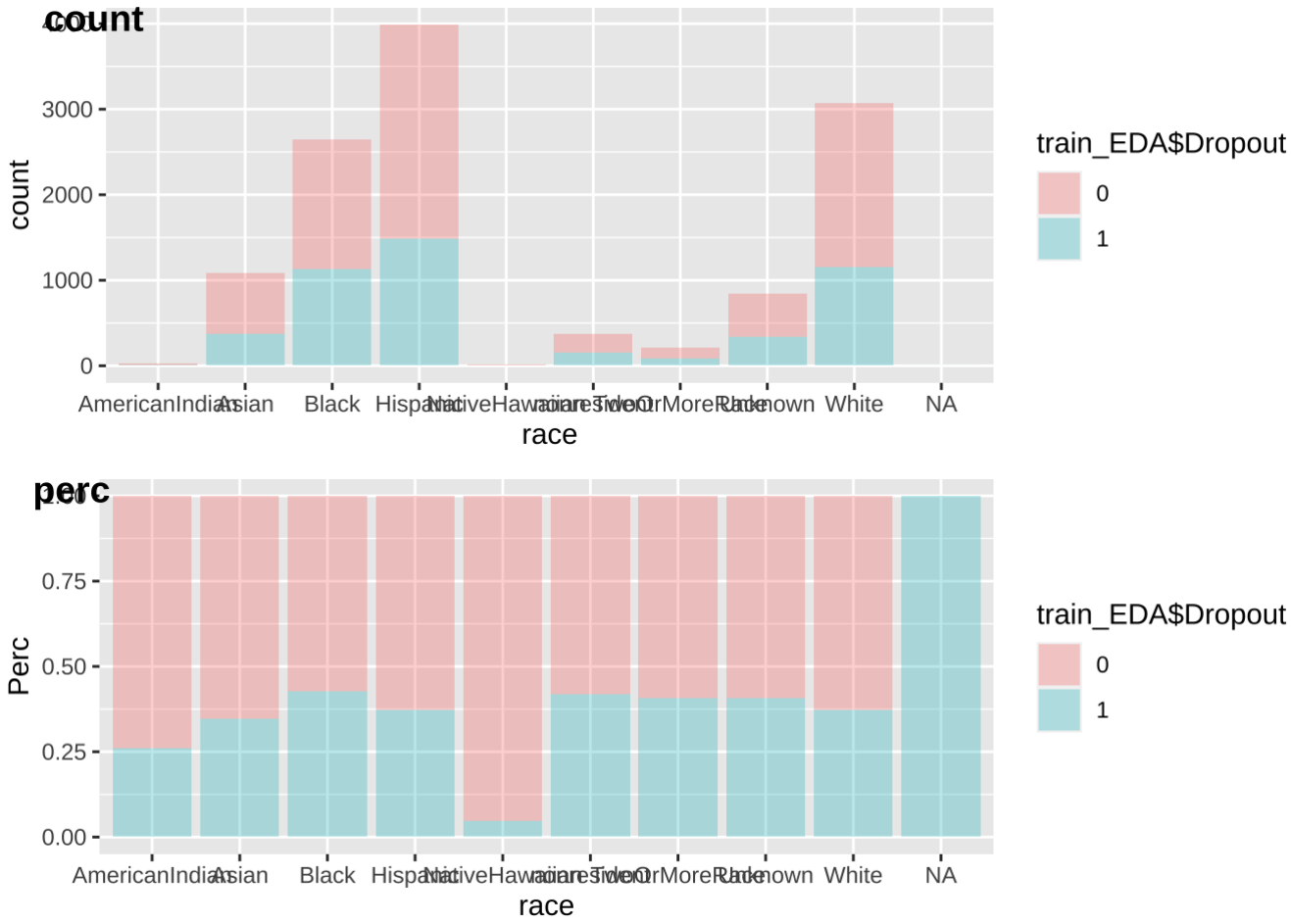

```
[1] "final_majorOne"
```

```
Warning in chisq.test(get(i)): Chi-squared approximation may be incorrect
```

```
Warning in chisq.test(get(i)): Chi-squared approximation may be incorrect
```




```
    Pearson's Chi-squared test

data:  get(i)
X-squared = 529.11, df = 18, p-value < 2.2e-16
```




```
[1] "race"

    Pearson's Chi-squared test

data:  get(i)
X-squared = 45.91, df = 8, p-value = 2.472e-07
```

```
print(drop)
```

```
[1] "Mother.s.Highest.Grade.Level" "Gender"
[3] "BirthMonth"                   "complete_DevMath"
```

Analysis and Main Result

During the chi square test, there are 4 variables having p-value>0.001, which means they are not significant at 0.001 level, so we have to delete them. These four are: `Mother.s.Highest.Grade.Level`, `Gender`, `BirthMonth`, and `Complete_Devmath`.

Involved variables:

- Marital.Status,
- Father.s.Highest.Grade.Level,

- Housing,
- EnrollmentStatus,
- HighDeg,
- MathPlacement,
- EngPlacement,
- GatewayMathStatus,
- GatewayEnglishStatus,
- complete_DevEnglish,
- final_Complete1,
- first_term_Major1,
- final_majorOne,
- race

3.3 Conlusions based on EDA

Involved variables:

```
df<-df %>% select(-c(Mother.s.Highest.Grade.Level,Gender,BirthMonth))
colnames(df)
```

```
 [1] "StudentID"                   "cohort"
 [3] "cohort.term"                 "Marital.Status"
 [5] "Adjusted.Gross.Income"       "Parent.Adjusted.Gross.Income"
 [7] "Father.s.Highest.Grade.Level" "Housing"
 [9] "RegistrationDate"            "BirthYear"
[11] "EnrollmentStatus"            "NumColCredAttemptTransfer"
[13] "NumColCredAcceptTransfer"    "HighDeg"
[15] "MathPlacement"               "EngPlacement"
[17] "GatewayMathStatus"           "GatewayEnglishStatus"
[19] "complete_DevMath"            "complete_DevEnglish"
[21] "final_Complete1"             "final_GPA"
[23] "cohort_year"                 "first_term_Major1"
[25] "final_majorOne"              "total_Loan"
[27] "total_Scholarship"           "total_Work_Study"
[29] "total_Grant"                 "race"
[31] "overall_income"
```

```
train_AFTER_EDA <- left_join(dropput_train, df, by="StudentID")
test_AFTER_EDA <- left_join(test_id, df, by="StudentID")
train_AFTER_EDA$Dropout <- factor(train_EDA$Dropout)
de<-which(is.na(train_AFTER_EDA$"BirthYear"),arr.ind = TRUE)
train_AFTER_EDA<-train_AFTER_EDA[-de,]
train_AFTER_EDA <- select(train_AFTER_EDA,-StudentID)


# write.csv(train_AFTER_EDA,file="output data/train_AFTER_EDA.csv",row.names = FALSE)
# write.csv(test_AFTER_EDA,file="output data/test_AFTER_EDA.csv",row.names = FALSE)
# write.csv(df,file="output data/df_AFTER_EDA.csv",row.names = FALSE)
```

4.1 Decision Trees

Decision trees used tree-based partitions to handle both continuous and categorical variables, so it can provide good classification for the data set we are working on. But regression tree can only work on ordered continuous value, which is limited.

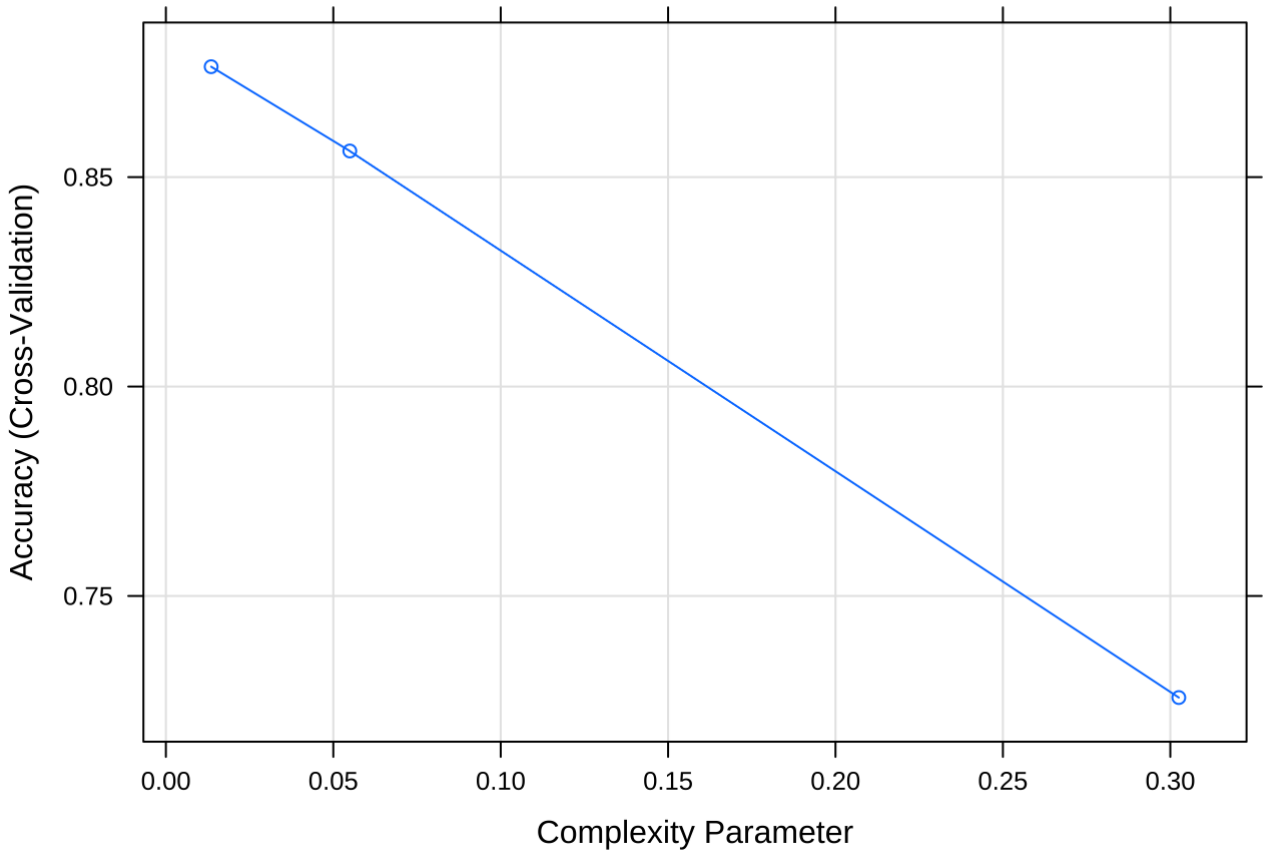
```
trainEDA<-train_AFTER_EDA
testEDA<-test_AFTER_EDA
trainEDA$Dropout <- as.factor(trainEDA$Dropout)
library(tidyverse)
library(caret)
set.seed(12345)
intrain <- createDataPartition(trainEDA$Dropout, p=0.75, list = FALSE)
train <- trainEDA[intrain,]
test <- trainEDA[-intrain,]

trctrl <- trainControl(method = "cv",
                       number = 10
)

tree_model <- train(Dropout ~ ., method='rpart', data=train, trControl = trctrl)
print(tree_model$finalModel)
```

```
n= 9196

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 9196 3550 0 (0.61396259 0.38603741)
   2) final_Complete1>=3.5 2646   78 0 (0.97052154 0.02947846) *
   3) final_Complete1< 3.5 6550 3078 1 (0.46992366 0.53007634)
     6) RegistrationDate>=2.015013e+07 2646  446 0 (0.83144369 0.16855631) *
     7) RegistrationDate< 2.015013e+07 3904  878 1 (0.22489754 0.77510246)
      14) total_Grant>=4.393268 545  175 0 (0.67889908 0.32110092) *
      15) total_Grant< 4.393268 3359  508 1 (0.15123549 0.84876451) *
```

```
tree_model$bestTune
```

```
          cp
1 0.01352113
```

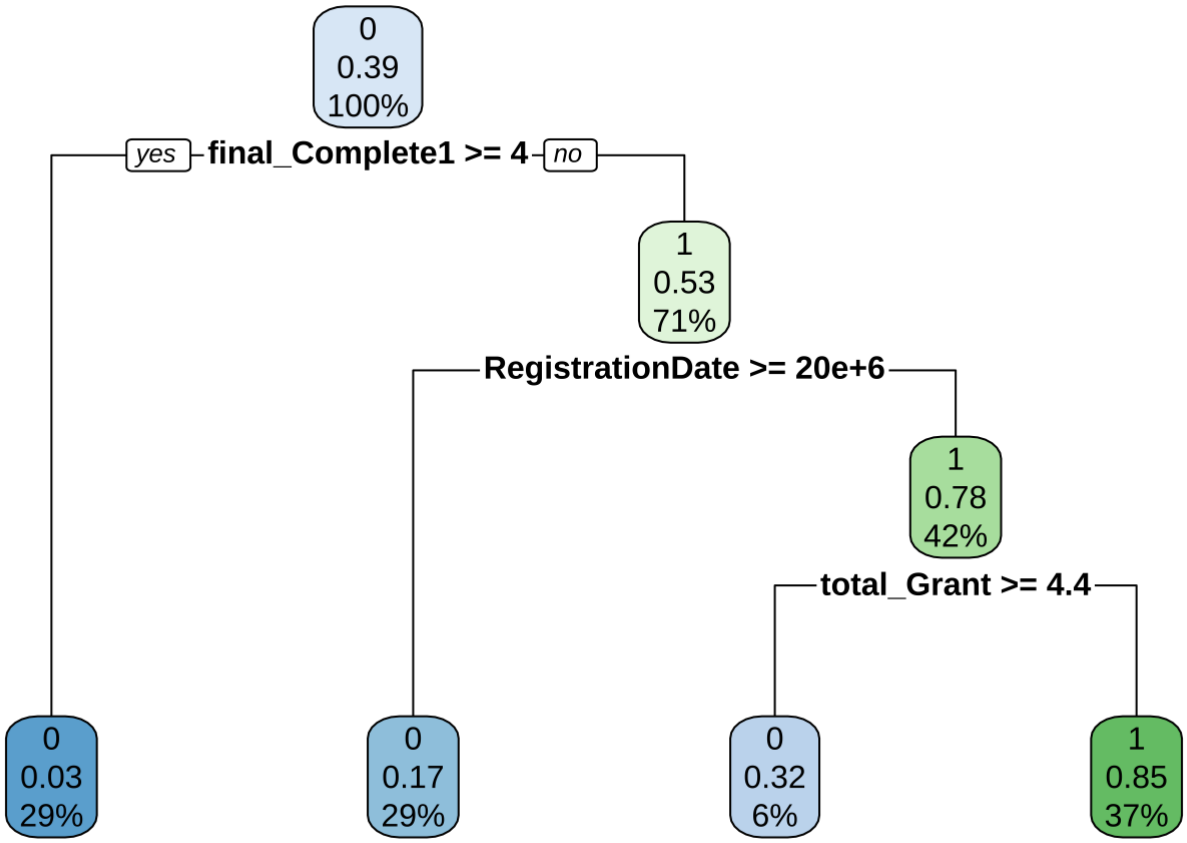
```
#Plot complexity parameter tuning runs
plot(tree_model)
```




```
#Plot the tree
library(rpart.plot)
```

```
Loading required package: rpart
```

```
rpart.plot(tree_model$finalModel)
```




```
#Predict
predictions <- predict(tree_model, newdata = test, type='raw')
#Model performance
confusionMatrix(predictions,test$Dropout)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1732  259
         1  149  924

               Accuracy : 0.8668
                 95% CI : (0.8543, 0.8787)
    No Information Rate : 0.6139
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7142

 Mcnemar's Test P-Value : 6.803e-08

            Sensitivity : 0.9208
            Specificity : 0.7811
```

```
         Pos Pred Value : 0.8699
         Neg Pred Value : 0.8611
             Prevalence : 0.6139
         Detection Rate : 0.5653
   Detection Prevalence : 0.6498
      Balanced Accuracy : 0.8509

       'Positive' Class : 0
```

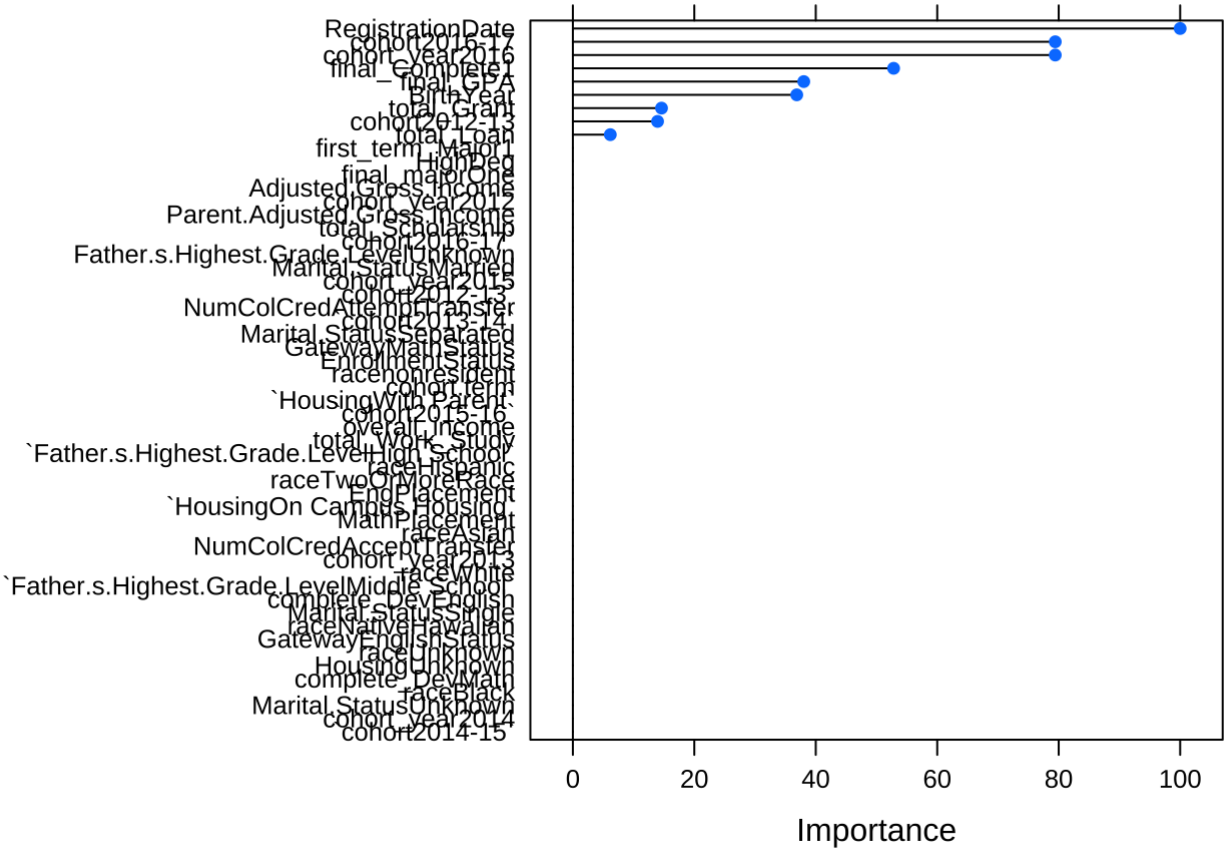
```
# plot(forestImp)
F_meas(predictions,test$Dropout)
```

```
[1] 0.8946281
```

```
#To see the importance of the variables
treeImp <- varImp(tree_model, scale = TRUE)
treeImp
```

```
rpart variable importance

  only 20 most important variables shown (out of 54)

                            Overall
RegistrationDate            100.000
cohort_year2016              79.432
cohort2016-17                79.432
final_Complete1              52.820
final_GPA                    38.026
BirthYear                    36.871
total_Grant                  14.601
cohort2012-13                13.950
total_Loan                    6.168
GatewayMathStatus             0.000
`HousingWith Parent`          0.000
raceBlack                     0.000
raceHispanic                  0.000
HousingUnknown                0.000
complete_DevEnglish           0.000
cohort_year2013               0.000
Adjusted.Gross.Income         0.000
`cohort2012-13`               0.000
Parent.Adjusted.Gross.Income  0.000
cohort_year2015               0.000
```

```
plot(treeImp)
```




```
summary(treeImp)
```

```
           Length Class      Mode
importance 1      data.frame list
model      1      -none-     character
calledFrom 1      -none-     character
```

```
precision(predictions,test$Dropout)
```

```
[1] 0.8699146
```

```
recall(predictions,test$Dropout)
```

```
[1] 0.9207868
```

4.2 Logistic

```
library(tidyverse)
library(caret)
library(ggplot2)
library(GGally)
```

```
library(GGally)
library(ISLR2)
library(pROC)
```

```
Type 'citation("pROC")' for a citation.


Attaching package: 'pROC'

The following objects are masked from 'package:stats':

    cov, smooth, var
```

```
library(car)
```

```
Loading required package: carData


Attaching package: 'car'

The following object is masked from 'package:dplyr':

    recode

The following object is masked from 'package:purrr':

    some
```

```
set.seed(12345)
train <- train_AFTER_EDA
train$Dropout <- factor(ifelse(train$Dropout == 1, "Yes", "No"))
train <- train %>% select(-cohort)
```

call Confusion Matrix to get F1

```
call <- function(cm) {
  print(cm$byClass["Sensitivity"])
  print(cm$byClass["Specificity"])
  print(cm$byClass["Precision"])
  print(cm$byClass["Recall"])
  print(cm$byClass["F1"])
  print(cm$overall["Accuracy"])
}
```

set 10 cross validation and partition

```
trctrl <- trainControl(method = "cv", number = 10, classProbs = T)
intrain <- createDataPartition(train$Dropout, p = 0.75, list = FALSE)
trainModel <- train[intrain, ]
testModel <- train[-intrain, ]
```

step 1 try logistic regression for all variables

```
logicModel <- train(Dropout ~.,
                    method = "glm",
                    preProcess = c("scale", "center"),
                    data = trainModel,
                    trControl = trctrl,
                    family = binomial(link = "logit"))
```

```
Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(logicModel)
```

```
Call:
NULL

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.2431  -0.2822  -0.0001   0.3515   3.8529

Coefficients:
```

	Estimate	Std. Error	z value
(Intercept)	-3.673517	37.048239	-0.099
cohort.term	-0.103854	0.046302	-2.243
Marital.StatusMarried	-0.113305	0.093053	-1.218
Marital.StatusSeparated	-0.006659	0.053567	-0.124
Marital.StatusSingle	-0.137030	0.146338	-0.936
Marital.StatusUnknown	0.007442	0.294793	0.025
Adjusted.Gross.Income	0.013827	0.029168	0.474
Parent.Adjusted.Gross.Income	-0.238802	0.057212	-4.174
`Father.s.Highest.Grade.LevelHigh School`	0.042463	0.047610	0.892
`Father.s.Highest.Grade.LevelMiddle School`	0.032681	0.042996	0.760
Father.s.Highest.Grade.LevelUnknown	0.002545	0.057688	0.044
`HousingOn Campus Housing`	0.038722	0.044823	0.864
HousingUnknown	-0.273979	0.266214	-1.029
`HousingWith Parent`	-0.073217	0.047120	-1.554
RegistrationDate	-0.099414	0.284828	-0.349
BirthYear	-0.055635	0.057218	-0.972
EnrollmentStatus	-0.514889	0.086875	-5.927
NumColCredAttemptTransfer	0.078145	0.079098	0.988
NumColCredAcceptTransfer	0.135176	0.100096	1.350
HighDeg	0.093676	0.050911	1.840
MathPlacement	-0.092765	0.073196	-1.267
EngPlacement	-0.269416	0.067185	-4.010
GatewayMathStatus	-0.127688	0.041965	-3.043
GatewayEnglishStatus	-0.121215	0.049785	-2.435
complete_DevMath	-0.311277	0.057654	-5.399
complete_DevEnglish	-0.111355	0.059492	-1.872
final_Complete1	-2.435947	0.069489	-35.055
final_GPA	-0.966626	0.051095	-18.918
cohort_year2012	-0.206613	0.083388	-2.478
cohort_year2013	-0.532813	0.133745	-3.984
cohort_year2014	-0.831078	0.195661	-4.248
cohort_year2015	-1.426339	0.259310	-5.501
cohort_year2016	-8.315055	87.216189	-0.095
first_term_Major1	-0.585815	1.254519	-0.467
final_majorOne	0.590434	1.254563	0.471
total_Loan	-0.453518	0.045299	-10.012
total_Scholarship	-0.409685	0.045272	-9.050
total_Work_Study	-0.141571	0.036384	-3.891
total_Grant	-0.517623	0.056871	-9.102
raceAsian	0.282269	0.221405	1.275
raceBlack	0.485174	0.320228	1.515
raceHispanic	0.458972	0.363950	1.261
raceNativeHawaiian	-0.688242	88.455667	-0.008
racenonresident	0.129446	0.134788	0.960
raceTwoOrMoreRace	0.175955	0.108872	1.616
raceUnknown	0.320951	0.196606	1.632
raceWhite	0.530038	0.337931	1.568
overall_income	-0.049044	0.062794	-0.781
	Pr(>|z|)		
(Intercept)	[illegible]		

```
(Intercept)                                 0.92102
cohort.term                                 0.02490 *
Marital.StatusMarried                       0.22336
Marital.StatusSeparated                     0.90107
Marital.StatusSingle                        0.34907
Marital.StatusUnknown                       0.97986
Adjusted.Gross.Income                       0.63546
Parent.Adjusted.Gross.Income               2.99e-05 ***
`Father.s.Highest.Grade.LevelHigh School`   0.37245
`Father.s.Highest.Grade.LevelMiddle School` 0.44720
Father.s.Highest.Grade.LevelUnknown         0.96481
`HousingOn Campus Housing`                  0.38765
HousingUnknown                              0.30340
`HousingWith Parent`                        0.12022
RegistrationDate                            0.72707
BirthYear                                   0.33089
EnrollmentStatus                           3.09e-09 ***
NumColCredAttemptTransfer                   0.32317
NumColCredAcceptTransfer                    0.17686
HighDeg                                     0.06577 .
MathPlacement                               0.20503
EngPlacement                               6.07e-05 ***
GatewayMathStatus                           0.00234 **
GatewayEnglishStatus                        0.01490 *
complete_DevMath                           6.70e-08 ***
complete_DevEnglish                         0.06124 .
final_Complete1                             < 2e-16 ***
final_GPA                                   < 2e-16 ***
cohort_year2012                             0.01322 *
cohort_year2013                            6.78e-05 ***
cohort_year2014                            2.16e-05 ***
cohort_year2015                            3.79e-08 ***
cohort_year2016                             0.92405
first_term_Major1                           0.64053
final_majorOne                              0.63791
total_Loan                                  < 2e-16 ***
total_Scholarship                           < 2e-16 ***
total_Work_Study                           9.98e-05 ***
total_Grant                                 < 2e-16 ***
raceAsian                                   0.20234
raceBlack                                   0.12975
raceHispanic                                0.20728
raceNativeHawaiian                          0.99379
racenonresident                             0.33687
raceTwoOrMoreRace                           0.10606
raceUnknown                                 0.10258
raceWhite                                   0.11677
overall_income                              0.43479
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 12266.4  on 9195  degrees of freedom
Residual deviance:  4796.2  on 9148  degrees of freedom
AIC: 4892.2

Number of Fisher Scoring iterations: 18
```

```r
logic_pred <- predict(logicModel, newdata = testModel)
cm <- confusionMatrix(logic_pred, testModel$Dropout)
call(cm)
```

```
Sensitivity
   0.917597
Specificity
  0.8664413
Precision
0.9161359
  Recall
0.917597
       F1
0.9168659
Accuracy
0.897846
```

step 2 delete some variables that can nearly linear

Because there are many warnings in the console, so I decided to use some ways to eliminate some of the variables.

```r
colnames(trainModel[nearZeroVar(trainModel)])
```

```
[1] "total_Scholarship" "total_Work_Study"
```

By finding variables that may have var tends to have zero, we want to delete those variables and run again the model. We want to delete `total_Scholarship`, `total_Work_Study`.

```r
logicModel <- train(Dropout ~. -total_Scholarship -total_Work_Study ,
                    method = "glm",
                    preProcess = c("scale", "center"),
                    data = trainModel,
                    trControl = trctrl,
                    family = binomial(link = "logit"))
```

```
Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(logicModel)
```

```
Call:
NULL

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.2275  -0.2814  -0.0001   0.3627   3.8701

Coefficients:
                                          Estimate Std. Error z value
(Intercept)                             -3.650397  37.224731  -0.098
cohort.term                             -0.067456   0.045663  -1.477
Marital.StatusMarried                   -0.107544   0.091527  -1.175
Marital.StatusSeparated                  0.005089   0.052591   0.097
Marital.StatusSingle                    -0.107504   0.144050  -0.746
Marital.StatusUnknown                    0.021462   0.291681   0.074
Adjusted.Gross.Income                    0.014350   0.029252   0.491
Parent.Adjusted.Gross.Income            -0.300934   0.055995  -5.374
`Father.s.Highest.Grade.LevelHigh School`    0.038882   0.047152   0.825
`Father.s.Highest.Grade.LevelMiddle School`  0.033727   0.042184   0.800
Father.s.Highest.Grade.LevelUnknown      0.002363   0.057071   0.041
`HousingOn Campus Housing`               0.009988   0.044127   0.226
HousingUnknown                          -0.174163   0.263293  -0.661
`HousingWith Parent`                    -0.059194   0.046543  -1.272
RegistrationDate                        -0.112720   0.281882  -0.400
BirthYear                               -0.073920   0.056200  -1.315
EnrollmentStatus                        -0.395952   0.084961  -4.660
NumColCredAttemptTransfer                0.090928   0.078727   1.155
NumColCredAcceptTransfer                 0.124041   0.099905   1.242
HighDeg                                  0.085207   0.050417   1.690
```

```
MathPlacement                                -0.123052   0.070657  -1.742
EngPlacement                                 -0.229100   0.065484  -3.499
GatewayMathStatus                            -0.127224   0.041329  -3.078
GatewayEnglishStatus                         -0.115063   0.049130  -2.342
complete_DevMath                             -0.241685   0.055175  -4.380
complete_DevEnglish                          -0.087552   0.058973  -1.485
final_Complete1                              -2.425300   0.068786 -35.259
final_GPA                                    -1.054661   0.050633 -20.829
cohort_year2012                              -0.186290   0.082416  -2.260
cohort_year2013                              -0.512896   0.132062  -3.884
cohort_year2014                              -0.799260   0.193431  -4.132
cohort_year2015                              -1.372092   0.256430  -5.351
cohort_year2016                              -8.265513  87.632325  -0.094
first_term_Major1                            -0.615374   1.261611  -0.488
final_majorOne                                0.613672   1.261638   0.486
total_Loan                                   -0.407206   0.044267  -9.199
total_Grant                                  -0.551188   0.055921  -9.857
raceAsian                                     0.287590   0.219347   1.311
raceBlack                                     0.475850   0.317211   1.500
raceHispanic                                  0.468232   0.360525   1.299
raceNativeHawaiian                           -0.685297  88.809797  -0.008
racenonresident                               0.113526   0.133447   0.851
raceTwoOrMoreRace                             0.163762   0.107678   1.521
raceUnknown                                   0.316286   0.194736   1.624
raceWhite                                     0.525957   0.334697   1.571
overall_income                               -0.021521   0.062138  -0.346
                                             Pr(>|z|)
(Intercept)                                  0.921882
cohort.term                                  0.139612
Marital.StatusMarried                        0.239995
Marital.StatusSeparated                      0.922914
Marital.StatusSingle                         0.455487
Marital.StatusUnknown                        0.941344
Adjusted.Gross.Income                        0.623737
Parent.Adjusted.Gross.Income                 7.69e-08 ***
`Father.s.Highest.Grade.LevelHigh School`    0.409595
`Father.s.Highest.Grade.LevelMiddle School`  0.423988
Father.s.Highest.Grade.LevelUnknown          0.966974
`HousingOn Campus Housing`                   0.820940
HousingUnknown                               0.508304
`HousingWith Parent`                         0.203441
RegistrationDate                             0.689242
BirthYear                                    0.188409
EnrollmentStatus                             3.16e-06 ***
NumColCredAttemptTransfer                    0.248103
NumColCredAcceptTransfer                     0.214388
HighDeg                                      0.091019 .
MathPlacement                                0.081587 .
EngPlacement                                 0.000468 ***
GatewayMathStatus                            0.002082 **
```

```
GatewayEnglishStatus                  0.019181 *
complete_DevMath                      1.18e-05 ***
complete_DevEnglish                   0.137650
final_Complete1                        < 2e-16 ***
final_GPA                              < 2e-16 ***
cohort_year2012                       0.023799 *
cohort_year2013                       0.000103 ***
cohort_year2014                       3.60e-05 ***
cohort_year2015                       8.76e-08 ***
cohort_year2016                       0.924855
first_term_Major1                     0.625714
final_majorOne                        0.626677
total_Loan                             < 2e-16 ***
total_Grant                            < 2e-16 ***
raceAsian                             0.189818
raceBlack                             0.133587
raceHispanic                          0.194030
raceNativeHawaiian                    0.993843
racenonresident                       0.394925
raceTwoOrMoreRace                     0.128299
raceUnknown                           0.104337
raceWhite                             0.116080
overall_income                        0.729086
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 12266  on 9195  degrees of freedom
Residual deviance:  4899  on 9150  degrees of freedom
AIC: 4991

Number of Fisher Scoring iterations: 18
```

```r
logic_pred <- predict(logicModel, newdata = testModel)
cm <- confusionMatrix(logic_pred, testModel$Dropout)
call(cm)
```

```
Sensitivity
  0.9117491
Specificity
  0.8596788
Precision
0.9117491
   Recall
0.9117491
       F1
0.9117491
 Accuracy
0.8916449
```

We tried logistic regression as it is a classical way to find the probability that separate binary data. During the regression process, we have tried eliminate some of the variables that are nearly linear.

We have also used `vif` to try to eliminate some variables that are variance-inflated, but the result is not good.

We have also tried lasso and ridge to get rid of Multicollinearity, but it also seems to have a less F1.

Performance metrics

Accuracy for testing data: 0.8870757

Precision for testing data: 0.8937917

Recall rate for testing data: 0.9261031

F1 score for testing data: 0.9096606

4.3 Naive Bayes

```
library(readr)
library(tidyverse)
library(caret)
library(ggplot2)
library(lattice)
library(klaR)
```

```
Loading required package: MASS


Attaching package: 'MASS'

The following object is masked from 'package:ISLR2':

    Boston

The following object is masked from 'package:dplyr':

    select
```

```
set.seed(12345)
intrain <- createDataPartition(train_AFTER_EDA$Dropout,p=0.75,list = FALSE)
train <- train_AFTER_EDA[intrain,]
test <- train_AFTER_EDA[-intrain,]

trctrl <- trainControl(method = "cv",
                       number = 10,
                       classProbs = FALSE,
                       )
```

```
train[sapply(train, is.character)] <- lapply(train[sapply(train, is.character)],
                                             as.factor)
# Naive Bayes
nb_fit <- train(Dropout ~ Parent.Adjusted.Gross.Income+MathPlacement+EngPlacement+Gateway
                trControl=trctrl)
```

```
Warning: model fit failed for Fold01: usekernel=FALSE, fL=0, adjust=1 Error in
NaiveBayes.default(x, y, usekernel = FALSE, fL = param$fL, ...) :
  Zero variances for at least one class in variables: raceNativeHawaiian

Warning: model fit failed for Fold02: usekernel=FALSE, fL=0, adjust=1 Error in
NaiveBayes.default(x, y, usekernel = FALSE, fL = param$fL, ...) :
  Zero variances for at least one class in variables: raceNativeHawaiian

Warning: model fit failed for Fold03: usekernel=FALSE, fL=0, adjust=1 Error in
NaiveBayes.default(x, y, usekernel = FALSE, fL = param$fL, ...) :
  Zero variances for at least one class in variables: raceNativeHawaiian

Warning: model fit failed for Fold04: usekernel=FALSE, fL=0, adjust=1 Error in
NaiveBayes.default(x, y, usekernel = FALSE, fL = param$fL, ...) :
  Zero variances for at least one class in variables: raceNativeHawaiian

Warning: model fit failed for Fold05: usekernel=FALSE, fL=0, adjust=1 Error in
NaiveBayes.default(x, y, usekernel = FALSE, fL = param$fL, ...) :
  Zero variances for at least one class in variables: raceNativeHawaiian

Warning: model fit failed for Fold06: usekernel=FALSE, fL=0, adjust=1 Error in
NaiveBayes.default(x, y, usekernel = FALSE, fL = param$fL, ...) :
  Zero variances for at least one class in variables: raceNativeHawaiian

Warning: model fit failed for Fold07: usekernel=FALSE, fL=0, adjust=1 Error in
NaiveBayes.default(x, y, usekernel = FALSE, fL = param$fL, ...) :
  Zero variances for at least one class in variables: raceNativeHawaiian

Warning: model fit failed for Fold08: usekernel=FALSE, fL=0, adjust=1 Error in
NaiveBayes.default(x, y, usekernel = FALSE, fL = param$fL, ...) :
  Zero variances for at least one class in variables: raceNativeHawaiian

Warning: model fit failed for Fold09: usekernel=FALSE, fL=0, adjust=1 Error in
NaiveBayes.default(x, y, usekernel = FALSE, fL = param$fL, ...) :
  Zero variances for at least one class in variables: raceNativeHawaiian

Warning: model fit failed for Fold10: usekernel=FALSE, fL=0, adjust=1 Error in
NaiveBayes.default(x, y, usekernel = FALSE, fL = param$fL, ...) :
  Zero variances for at least one class in variables: raceNativeHawaiian

Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
There were missing values in resampled performance measures.
```

```
Warning in train.default(x, y, weights = w, ...): missing values found in
aggregated results
```

```
nb_fit
```

```
Naive Bayes

9196 samples
  12 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 8276, 8277, 8276, 8276, 8276, 8276, ...
Resampling results across tuning parameters:

  usekernel  Accuracy   Kappa
  FALSE            NaN        NaN
   TRUE      0.6178771  0.0138093

Tuning parameter 'fL' was held constant at a value of 0
Tuning
 parameter 'adjust' was held constant at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were fL = 0, usekernel = TRUE and adjust
 = 1.
```

```
test$Dropout <- factor(test$Dropout)
class_prob <- predict(nb_fit, newdata = test, type="prob")
class_prob1 <- predict(nb_fit, newdata = test, type="raw")
#Report Accuracy, Precision, Recall rate, and F measure
confusionMatrix(class_prob1,test$Dropout)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1879 1174
         1    2    9

               Accuracy : 0.6162
                 95% CI : (0.5987, 0.6335)
    No Information Rate : 0.6139
    P-Value [Acc > NIR] : 0.4052

                  Kappa : 0.008

 Mcnemar's Test P-Value : <2e-16
```

```
            Sensitivity : 0.998937
            Specificity : 0.007608
         Pos Pred Value : 0.615460
         Neg Pred Value : 0.818182
             Prevalence : 0.613903
         Detection Rate : 0.613251
   Detection Prevalence : 0.996410
      Balanced Accuracy : 0.503272

       'Positive' Class : 0
```

```
F_meas(class_prob1,test$Dropout)
```

```
[1] 0.7616538
```

```
precision(class_prob1,test$Dropout)
```

```
[1] 0.6154602
```

```
recall(class_prob1,test$Dropout)
```

```
[1] 0.9989367
```

Performance Metrics

Accuracy for testing data: 0.7428

Sensitivity for testing data: 0.9048

Specificity for testing data: 0.4852

Precision for testing data: 0.7364777

Recall rate for testing data: 0.9048379

F-measure score for testing data: 0.8120229

Why Use Naive Bayes Model & Conclusion

- It is good at solving classification problems, mainly used for a high-dimensional training dataset
- It handles both discrete and continuous data, and highly scalable with the number of predictors and data points

4.4 Random Forest

Similar mechanism with decision tree to provide a good classification for continuous and categorical variables. But the random forest algorithm performs better in accuracy when predicting outcomes than the decision tree algorithm.

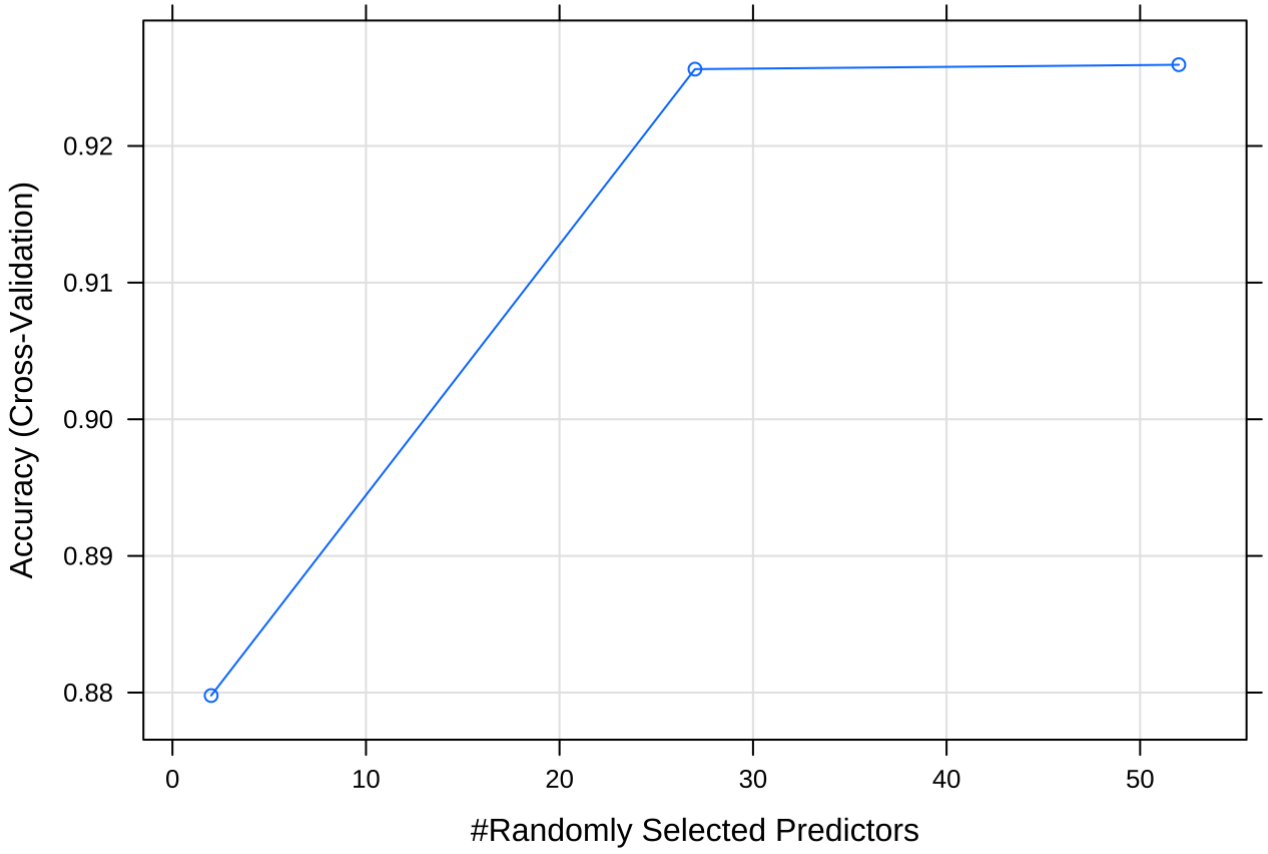
```
#Random Forest
library(caret)
set.seed(12345)
intrain <- createDataPartition(train_AFTER_EDA$Dropout,p=0.50,list = FALSE)
train1 <- train_AFTER_EDA[intrain,]
test1 <- train_AFTER_EDA[-intrain,]
#Cross validation
trctrl <- trainControl(method = "cv", number = 10)

forest_fit <- train(Dropout ~., data = train1, method = "rf",importance = T,
                    trControl=trctrl)
#To see model details
forest_fit
```

```
Random Forest

6131 samples
  30 predictor
   2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 5517, 5517, 5518, 5519, 5518, 5518, ...
Resampling results across tuning parameters:

  mtry  Accuracy   Kappa
   2    0.8797826  0.7382382
  27    0.9256220  0.8431774
  52    0.9259462  0.8440147

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was mtry = 52.
```

```
#To see the tuned mtry parameter.  Mtry is the number of randomly selected predictors
forest_fit$bestTune
```

```
  mtry
3   52
```

```
#Plot mtry parameter tuning runs
plot(forest_fit)
```

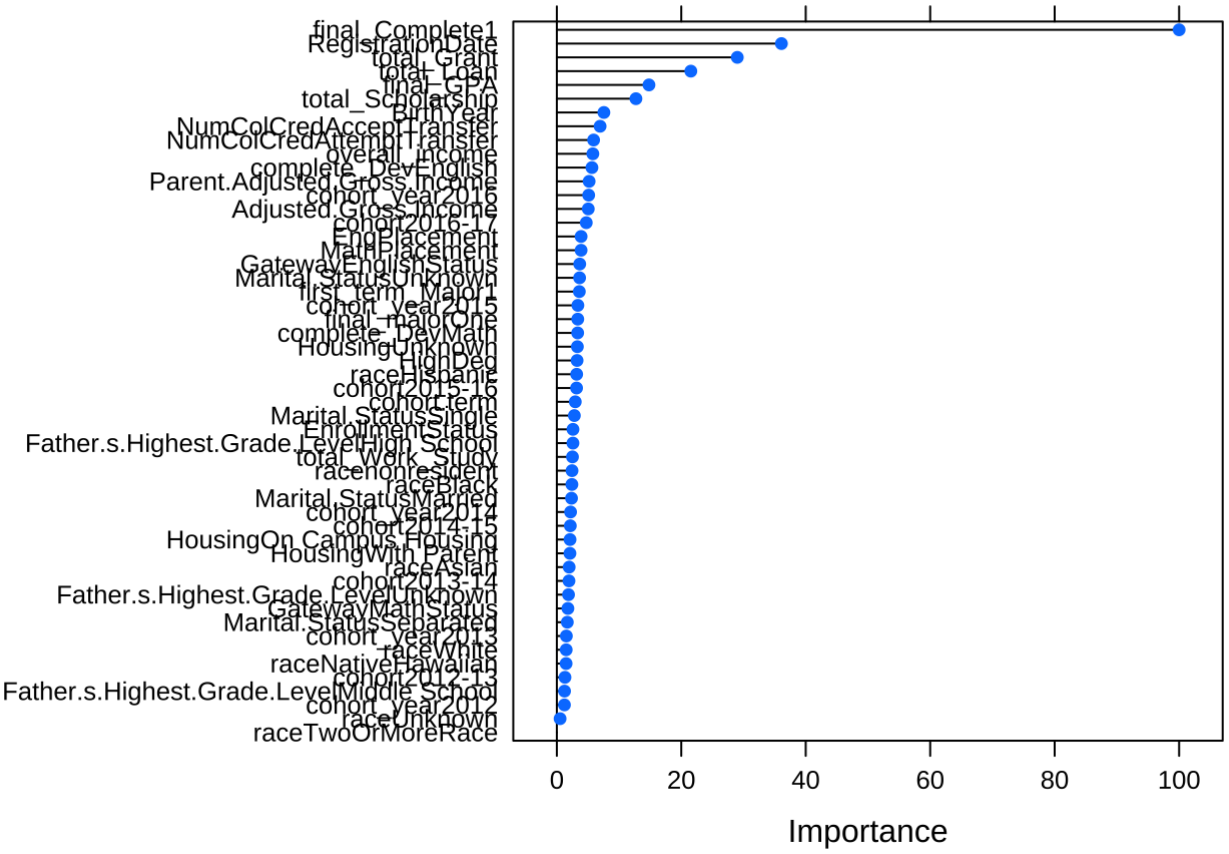



```
#Predict
predictionsf <- predict(forest_fit, newdata = test1)

#predictionsf <- predict(forest_fit, newdata = testEDA)
#Calculate performance measures
#postResample(forest_fit,train1$Dropout)
#To see the importance of the variables
forestImp <- varImp(forest_fit)
forestImp
```

```
rf variable importance

  only 20 most important variables shown (out of 52)

                           Importance
final_Complete1               100.000
RegistrationDate               36.097
total_Grant                    29.005
total_Loan                     21.539
final_GPA                      14.809
total_Scholarship              12.713
BirthYear                       7.564
NumColCredAcceptTransfer        6.954
```

```
NumColCredAttemptTransfer         5.908
overall_income                    5.788
complete_DevEnglish               5.640
Parent.Adjusted.Gross.Income      5.189
cohort_year2016                   5.118
Adjusted.Gross.Income             5.051
cohort2016-17                     4.728
EngPlacement                      3.899
MathPlacement                     3.897
GatewayEnglishStatus              3.678
Marital.StatusUnknown             3.658
first_term_Major1                 3.614
```

```
plot(forestImp)
```




```
#predictionsf <- predict(forest_fit, newdata = testEDA)
#Calculate performance measures
#postResample(forest_fit,testEDA$Dropout)

F_meas(predictionsf,test1$Dropout)
```

```
[1] 0.9373593
```

```
confusionMatrix(predictionsf,test1$Dropout)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 3539  249
         1  224 2117

               Accuracy : 0.9228
                 95% CI : (0.9159, 0.9294)
    No Information Rate : 0.614
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8369

 Mcnemar's Test P-Value : 0.2698

            Sensitivity : 0.9405
            Specificity : 0.8948
         Pos Pred Value : 0.9343
         Neg Pred Value : 0.9043
             Prevalence : 0.6140
         Detection Rate : 0.5774
   Detection Prevalence : 0.6180
      Balanced Accuracy : 0.9176

       'Positive' Class : 0
```

4.5 SVM

```
library(caret)
set.seed(12345)

#Create traning test split
intrain <- createDataPartition(train_AFTER_EDA$Dropout,p=0.75,list = FALSE)
train1 <- train_AFTER_EDA[intrain,]
test1 <- train_AFTER_EDA[-intrain,]

#Create cross validation
trctrl <- trainControl(method = "cv", number = 10)

call <- function(cm) {
print(cm$byClass["Sensitivity"])
print(cm$byClass["Specificity"])
print(cm$byClass["Precision"])
print(cm$byClass["Recall"])
```

```
print(cm$byClass["F1"])
print(cm$overall["Accuracy"])
}

# Linear kernel
modSVMFit_linear <- train(Dropout ~ .,method="svmLinear2", trControl=trctrl,data=train1)
#See model fit details
modSVMFit_linear$finalModel
```

```
Call:
svm.default(x = as.matrix(x), y = y, kernel = "linear", cost = param$cost,
    probability = classProbs)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  linear
       cost:  0.25

Number of Support Vectors:  2522
```

```
modSVMFit_linear$bestTune
```

```
  cost
1 0.25
```

```
SVMpredict_linear <- predict(modSVMFit_linear,test1)
confusionMatrix(SVMpredict_linear,test1$Dropout)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1720  164
         1  161 1019

               Accuracy : 0.8939
                 95% CI : (0.8825, 0.9046)
    No Information Rate : 0.6139
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7761

 Mcnemar's Test P-Value : 0.9117

            Sensitivity : 0.9144
            Specificity : 0.8614
         Pos Pred Value : 0.9130
```

```
         Neg Pred Value : 0.8636
             Prevalence : 0.6139
         Detection Rate : 0.5614
   Detection Prevalence : 0.6149
      Balanced Accuracy : 0.8879

       'Positive' Class : 0
```

```
SVMpredict_linear_matrix <- confusionMatrix(SVMpredict_linear,test1$Dropout)
F_meas(SVMpredict_linear,test1$Dropout)
```

```
[1] 0.9136786
```

```
call(SVMpredict_linear_matrix)
```

```
Sensitivity
  0.9144072
Specificity
  0.8613694
Precision
0.9129512
   Recall
0.9144072
       F1
0.9136786
 Accuracy
0.8939295
```

```
#Polynomial
#Use a tuning grid to tune parameters.  Need one column for each parameter that can be tu
modSVMFit_polynomial <- train(Dropout ~ .,method="svmPoly",trControl=trctrl,data=train1)
#See model fit details
modSVMFit_polynomial$finalModel
```

```
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 0.25

Polynomial kernel function.
 Hyperparameters : degree =  2  scale =  0.1  offset =  1

Number of Support Vectors : 2336

Objective Function Value : -459.3008
Training error : 0.07873
```

```
#The polynomial kernel is defined as (scale * crossprod(x, y) + offset)^degree
#See the tuning parametrs used (cost C, Scale of the kernel function)
modSVMFit_polynomial$bestTune
```

```
   degree scale    C
16      2   0.1 0.25
```

```
#Predict test dataset
SVMpredict_polynomial <- predict(modSVMFit_polynomial,test1)
confusionMatrix(SVMpredict_polynomial,test1$Dropout)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1742  166
         1  139 1017

               Accuracy : 0.9005
                 95% CI : (0.8893, 0.9108)
    No Information Rate : 0.6139
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7891

 Mcnemar's Test P-Value : 0.1366

            Sensitivity : 0.9261
            Specificity : 0.8597
         Pos Pred Value : 0.9130
         Neg Pred Value : 0.8798
             Prevalence : 0.6139
         Detection Rate : 0.5685
   Detection Prevalence : 0.6227
      Balanced Accuracy : 0.8929

       'Positive' Class : 0
```

```
SVMpredict_polynomial_matrix <- confusionMatrix(SVMpredict_polynomial,test1$Dropout)
F_meas(SVMpredict_polynomial,test1$Dropout)
```

```
[1] 0.9195038
```

```
call(SVMpredict_polynomial_matrix)
```

```
Sensitivity
  0.9261031
```

```
Specificity
  0.8596788
Precision
0.9129979
   Recall
0.9261031
       F1
0.9195038
 Accuracy
0.9004569
```

```
#Radial Kernel
modSVMFit_Radial <- train(Dropout ~ .,method="svmRadial",sigma =.2,trControl=trctrl,data=
#See model fit details
modSVMFit_Radial$finalModel
```

```
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 1

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  0.0116819268765185

Number of Support Vectors : 2865

Objective Function Value : -2292.432
Training error : 0.095476
```

```
#See the tuning parameters used (cost C, and sigma of the radial kernel function)
modSVMFit_Radial$bestTune
```

```
      sigma C
3 0.01168193 1
```

```
#Predict test dataset
SVMpredict_Radial <- predict(modSVMFit_Radial,test1)
confusionMatrix(SVMpredict_Radial,test1$Dropout)
```

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1722  159
         1  159 1024

               Accuracy : 0.8962
```

```
                95% CI : (0.8849, 0.9068)
    No Information Rate : 0.6139
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7811

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9155
            Specificity : 0.8656
         Pos Pred Value : 0.9155
         Neg Pred Value : 0.8656
             Prevalence : 0.6139
         Detection Rate : 0.5620
   Detection Prevalence : 0.6139
      Balanced Accuracy : 0.8905

       'Positive' Class : 0
```

```
SVMpredict_Radial_matrix <- confusionMatrix(SVMpredict_Radial,test1$Dropout)
call(SVMpredict_Radial_matrix)
```

```
Sensitivity
  0.9154705
Specificity
  0.8655959
Precision
0.9154705
   Recall
0.9154705
       F1
0.9154705
 Accuracy
0.8962141
```

We choose using SVM because we want to examine whether classes are nearly separable or not. Based on the correlation plot we made in EDA, we assumed that it is non-linear. For non-linear boundaries, kernel SVMs computations are less expensive than Logistic Regression.

Based on the results of three SVM Kernel Models, for both accuracy and F score, Polynomial Kernel has the best performance. As a result, the decision boundary is non-linear. However, since we need to pay more attention to students who will drop out and exercise necessary interventions, we need to focus on specificity (negative=1 in models). In that perspective, Radial Kernel performs the best.

4.6 XGBoost model

Library

Library

```
library(xgboost)
```

```
Attaching package: 'xgboost'

The following object is masked from 'package:dplyr':

    slice
```

```
library(tidyr)
library(Matrix)
```

```
Attaching package: 'Matrix'

The following objects are masked from 'package:tidyr':

    expand, pack, unpack
```

```
library(mlr)
```

```
Loading required package: ParamHelpers

Warning message: 'mlr' is in 'maintenance-only' mode since July 2019.
Future development will only happen in 'mlr3'
(<https://mlr3.mlr-org.com>). Due to the focus on 'mlr3' there might be
uncaught bugs meanwhile in {mlr} - please consider switching.
```

```
Attaching package: 'mlr'

The following object is masked from 'package:caret':

    train
```

```
library(Ckmeans.1d.dp)
library(ISLR2)
library(pROC)
```

```
Type 'citation("pROC")' for a citation.


Attaching package: 'pROC'

The following objects are masked from 'package:stats':

    cov, smooth, var
```

read data

```
DF_train<- read.csv("output data/train_AFTER_EDA.csv")
DF_test<- read.csv("output data/test_AFTER_EDA.csv")

#DF_train$Dropout<-factor(DF$Dropout,levels=c(0,1),labels=("no","yes"))
DF_train$Dropout <- factor(DF_train$Dropout)

DF_train$Marital.Status <- factor(DF_train$Marital.Status)
DF_train$Father.s.Highest.Grade.Level <- factor(DF_train$Father.s.Highest.Grade.Level)
DF_train$Housing <- factor(DF_train$Housing)
DF_train$race <- factor(DF_train$race)

DF_test$Marital.Status <- factor(DF_test$Marital.Status)
DF_test$Father.s.Highest.Grade.Level <- factor(DF_test$Father.s.Highest.Grade.Level)
DF_test$Housing <- factor(DF_test$Housing)
DF_test$race <- factor(DF_test$race)

#DF_train$Dropout <- factor(ifelse(DF_train$Dropout == 1, "Yes", "No"))
# DF_train<-DF_train%>%select(-c(StudentID,cohort))
# DF_test<-DF_test%>%select(-c(StudentID,cohort))
DF_train<-DF_train%>%select(-c(StudentID,cohort,RegistrationDate,final_Complete1,final_ma
DF_test<-DF_test%>%select(-c(StudentID,cohort,RegistrationDate,final_Complete1,final_majo
```

Split into training and testing

```
set.seed(12345)
intrain <- createDataPartition(DF_train$Dropout,p=0.75,list = FALSE)
```

```
intrain <- createDataPartition(DF_train$Dropout,p=0.75,list = FALSE)

train1 <- DF_train[intrain,]
test1 <- DF_train[-intrain,]
trctrl <- trainControl(method = "cv", number = 10)
```

Data preprocess

The xgboost function in the xgboost package requires a specific data format (xgb.DMatrix). Before using the xgboost function, we need to preprocess the dataset.

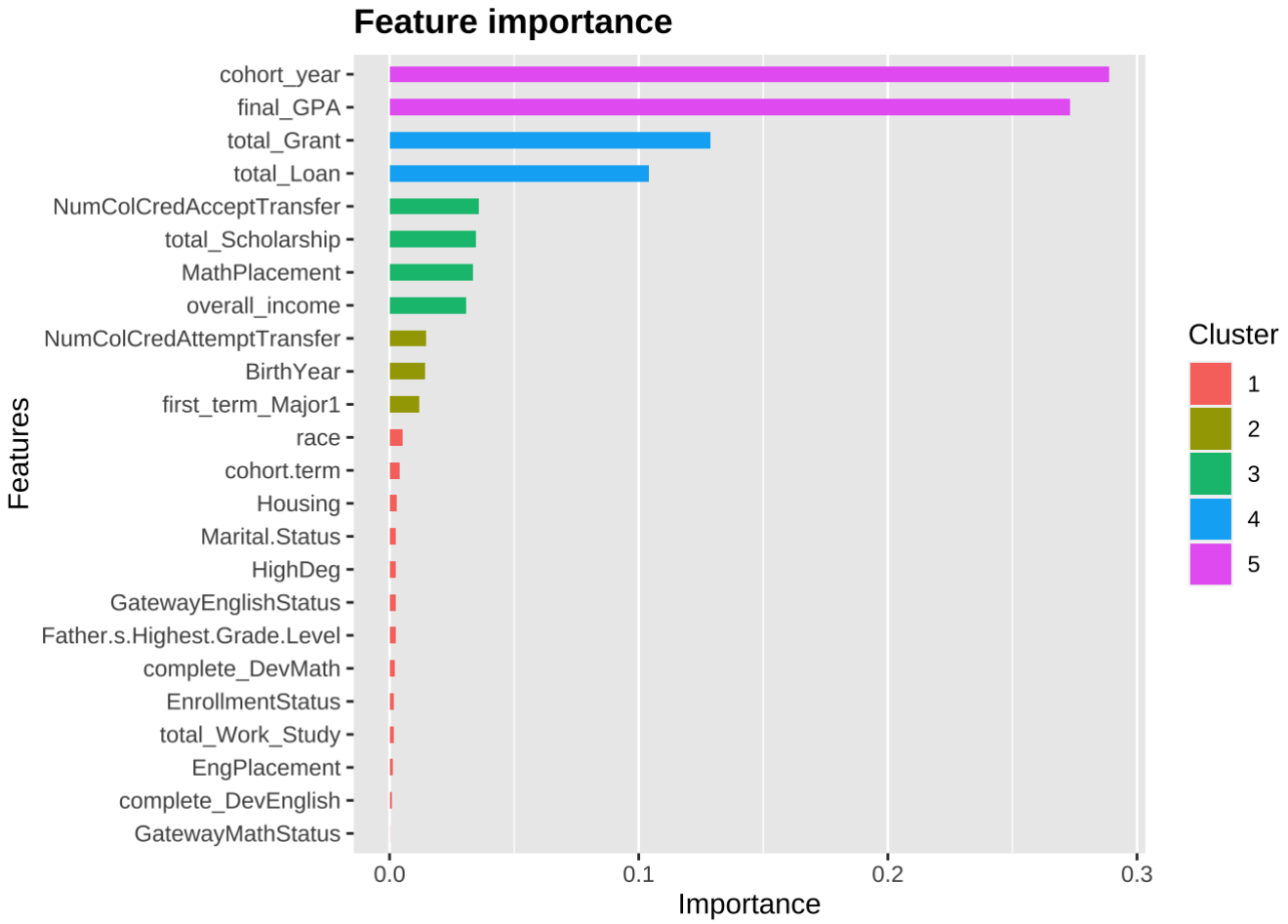
```
################################training set
#Convert (independent variable) in trainset to matrix
traindata1<-data.matrix(train1 %>% select(-c(Dropout)))
#Use the Matrix function to set the sparse parameter to TRUE and convert it to a sparse m
traindata2 <-Matrix(traindata1,sparse=T)

#Convert the dependent variable to numeric type.-1 is because we want to count from 0
train_y <-as.numeric(train1[,"Dropout"])-1
#Splice the independent variable and dependent variable into a list
traindata <-list(data=traindata2,label=train_y)

#The xgb.DMatrix required by the model is constructed from a sparse matrix
dtrain <-xgb.DMatrix(data=traindata$data,label=traindata$label)

################################testing set
#Convert (independent variable) in testset to matrix
testdata1<-data.matrix(test1 %>% select(-c(Dropout)))
#Use the Matrix function to set the sparse parameter to TRUE and convert it to a sparse m
testdata2 <-Matrix(testdata1,sparse=T)

#Convert the dependent variable to numeric type.-1 is because we want to count from 0
test_y <-as.numeric(test1[,"Dropout"])-1
#Splice the independent variable and dependent variable into a list
testdata <-list(data=testdata2,label=test_y)

#The xgb.DMatrix required by the model is constructed from a sparse matrix
dtest <-xgb.DMatrix(data=testdata$data,label=testdata$label)

################################final test data
#Convert (independent variable) in final test to matrix
final_test<-data.matrix(DF_test)
#Use the Matrix function to set the sparse parameter to TRUE and convert it to a sparse m
final_test <-Matrix(final_test,sparse=T)

#The xgb.DMatrix required by the model is constructed from a sparse matrix
final_test <-xgb.DMatrix(final_test)
```

fitting model and check performance

```
#fitting the xgboost model
xgb <-xgboost(data=dtrain,max_depth=6,eta=0.3,
nthread = 2,objective='binary:logistic',nround=25)
```

```
[1] train-logloss:0.562524
[2] train-logloss:0.488010
[3] train-logloss:0.441289
[4] train-logloss:0.410301
[5] train-logloss:0.387597
[6] train-logloss:0.363219
[7] train-logloss:0.348835
[8] train-logloss:0.337864
[9] train-logloss:0.327007
[10]    train-logloss:0.319131
[11]    train-logloss:0.309869
[12]    train-logloss:0.303535
[13]    train-logloss:0.296450
[14]    train-logloss:0.292285
[15]    train-logloss:0.286222
[16]    train-logloss:0.282901
[17]    train-logloss:0.275092
[18]    train-logloss:0.271358
[19]    train-logloss:0.267025
[20]    train-logloss:0.264758
[21]    train-logloss:0.261569
[22]    train-logloss:0.257379
[23]    train-logloss:0.251616
[24]    train-logloss:0.250596
[25]    train-logloss:0.249144
```

```
importance <- xgb.importance(model = xgb)
xgb.ggplot.importance(importance)
```




```
cv<-xgb.cv(data=dtrain,max_depth=6,eta=0.3,nfold = 5,metrics = list("rmse","auc"),nthread
```

```
[1] train-rmse:0.431423+0.000935    train-auc:0.902443+0.002821 test-
rmse:0.434792+0.001012 test-auc:0.886949+0.004484
[2] train-rmse:0.391249+0.000830    train-auc:0.915184+0.001852 test-
rmse:0.397562+0.002820 test-auc:0.898434+0.006366
[3] train-rmse:0.366900+0.001524    train-auc:0.920749+0.001800 test-
rmse:0.375719+0.004166 test-auc:0.903549+0.007983
[4] train-rmse:0.350125+0.001736    train-auc:0.926749+0.001597 test-
rmse:0.361971+0.004709 test-auc:0.907393+0.007642
[5] train-rmse:0.338678+0.001965    train-auc:0.930686+0.001635 test-
rmse:0.353106+0.005079 test-auc:0.910091+0.007518
[6] train-rmse:0.331256+0.002656    train-auc:0.933351+0.002109 test-
rmse:0.347921+0.005912 test-auc:0.911698+0.008153
[7] train-rmse:0.323625+0.002978    train-auc:0.937672+0.002162 test-
rmse:0.343114+0.006163 test-auc:0.914277+0.007786
[8] train-rmse:0.319077+0.003275    train-auc:0.939671+0.002470 test-
rmse:0.340179+0.006670 test-auc:0.915659+0.007327
[9] train-rmse:0.315041+0.002830    train-auc:0.941843+0.002034 test-
rmse:0.337693+0.006960 test-auc:0.917039+0.007141
[10]    train-rmse:0.311472+0.002599    train-auc:0.943988+0.002008 test-
rmse:0.335840+0.007847 test-auc:0.918209+0.007647
```

```
[11]    train-rmse:0.307958+0.003025    train-auc:0.946014+0.002102 test-
rmse:0.334385+0.007926 test-auc:0.919265+0.007428
[12]    train-rmse:0.304943+0.002391    train-auc:0.947738+0.001662 test-
rmse:0.333066+0.008114 test-auc:0.920109+0.007325
[13]    train-rmse:0.301207+0.002108    train-auc:0.950055+0.001445 test-
rmse:0.331690+0.008430 test-auc:0.920862+0.007463
[14]    train-rmse:0.299083+0.002291    train-auc:0.951388+0.001524 test-
rmse:0.330918+0.008293 test-auc:0.921454+0.007046
[15]    train-rmse:0.295635+0.002303    train-auc:0.953692+0.001389 test-
rmse:0.330189+0.008470 test-auc:0.922068+0.006978
[16]    train-rmse:0.293320+0.002493    train-auc:0.955140+0.001567 test-
rmse:0.329682+0.008604 test-auc:0.922323+0.006864
[17]    train-rmse:0.290178+0.002467    train-auc:0.957175+0.001455 test-
rmse:0.329187+0.008250 test-auc:0.922742+0.006469
[18]    train-rmse:0.287958+0.003310    train-auc:0.958459+0.001932 test-
rmse:0.328778+0.008280 test-auc:0.922974+0.006387
[19]    train-rmse:0.284987+0.002025    train-auc:0.960231+0.001041 test-
rmse:0.328497+0.008590 test-auc:0.923124+0.006715
[20]    train-rmse:0.283085+0.002339    train-auc:0.961359+0.001234 test-
rmse:0.327632+0.008701 test-auc:0.923852+0.006677
[21]    train-rmse:0.280786+0.002825    train-auc:0.962718+0.001562 test-
rmse:0.326871+0.008332 test-auc:0.924311+0.006252
[22]    train-rmse:0.278735+0.002383    train-auc:0.963962+0.001170 test-
rmse:0.327041+0.008316 test-auc:0.924201+0.006160
[23]    train-rmse:0.275743+0.003181    train-auc:0.965625+0.001564 test-
rmse:0.326683+0.008180 test-auc:0.924462+0.006058
[24]    train-rmse:0.273934+0.003117    train-auc:0.966557+0.001581 test-
rmse:0.326189+0.008075 test-auc:0.924908+0.005976
[25]    train-rmse:0.272286+0.002917    train-auc:0.967431+0.001476 test-
rmse:0.326030+0.008050 test-auc:0.924992+0.005988
```

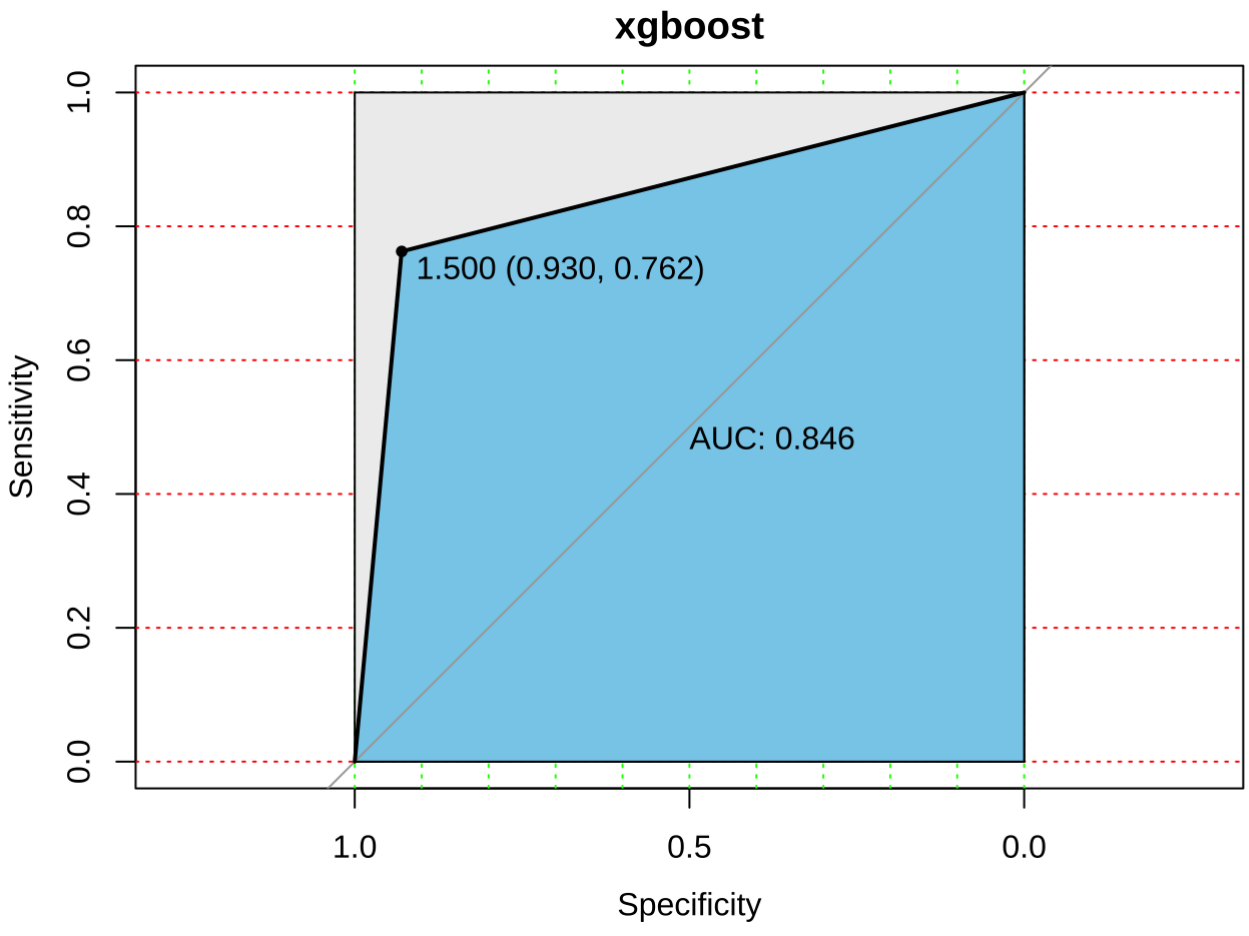
```r
#Make predictions on the test set
pre_xgb=round(predict(xgb,newdata=dtest))
pre_xgb_final=round(predict(xgb,newdata=final_test))

#Output confusion matrix
pre_xgb<-as.factor(pre_xgb)
cm<-confusionMatrix(pre_xgb,test1$Dropout,positive="0")

call <- function(cm) {
 print(cm$byClass["Sensitivity"])
 print(cm$byClass["Specificity"])
 print(cm$byClass["Precision"])
 print(cm$byClass["Recall"])
 print(cm$byClass["F1"])
 print(cm$overall["Accuracy"])
 }

call(cm)
```

```
Sensitivity
  0.9298246
Specificity
  0.7624683
Precision
0.8615764
   Recall
0.9298246
       F1
0.8944004
 Accuracy
0.8652089
```

```
#plot ROC curve and AUC value
xgboost_roc<-roc(test1$Dropout,as.numeric(pre_xgb))
```

```
Setting levels: control = 0, case = 1

Setting direction: controls < cases
```

```
plot(xgboost_roc,print.auc=TRUE,auc.polygon=TRUE,grid=c(0.1,0.2),grid.col=c("green","red"
```



Out put final prediction

```
df_test_clean <- read.csv("output data/test_AFTER_EDA.csv")
test_pred <- pre_xgb_final
df_test_clean <- df_test_clean %>%
  mutate(Dropout = test_pred)



submission <- data.frame(
     StudentID = df_test_clean$StudentID,
     Dropout = df_test_clean$Dropout
 )

#write.csv(submission,file="output data/submission_xgb.csv",row.names = FALSE)
```

conclusion:

XGBoost is an ensemble machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems(classification and regression) in a fast and accurate way and has been widely used in data competitions such as kaggel in recent years.

In this work, XGBoost was used to predict dropout, 25 variables are used to fit the model. For parameters, max_depth=6,eta=0.3,nthread = 2,objective='binary:logistic',nround=25 was used in the model and finally get an AUC with 0.846 and F1 with0.894. The model also ranked the top importance of variables which are cohort year final gpa, total grant, total loan, NumColCredAcceptTransfer.